# FAMILY COMPUTING

# DICTIONARY OF COMPUTER TERMS MADE SIMPLE

FAMILY COMPUTING

# DICTIONARY OF COMPUTER TERMS MADE SIMPLE

by
PATRICIA CONNIFFE

SCHOLASTIC INC.
New York Toronto London Auckland Sydney Tokyo

Editorial Director: Eleanor Angeles
Editor: Mollie L. Cohen
Pronunciation Key: Marjorie Burns

Design: Harry Chester Inc.
Illustrations: Robert Jackson

ISBN 0-590-94456-8

12 11 10 9 8 7 6 5 4 3 2 1 9 3 4 5 6/ 8
Printed in the U.S.A. 14

# INTRODUCTION TO THE COMPUTER DICTIONARY

It's already happened, hasn't it? You're looking through your friend's computer manual; or your teacher hands you a magazine about computers; or you hear someone talking about a new computer job . . . and terms you've never heard before keep popping up every few seconds. *Pixel, LOGO, semiconductor.* Some aren't even words! *CP/M, DOS, PL1:* It's like alphabet soup. You decide to check out a few of these terms in your family dictionary. As you keep turning pages, the truth slowly dawns. Most of these words aren't *in* the dictionary! There's no place to look up THESE WORDS!

As of now, that's all in the past. With Scholastic's illustrated *Computer Dictionary* you've got references to more than 500 terms that cover:

- **different kinds of computers, and how they work**—from mainframes, to minicomputers and micros; computers that add, send messages, remember, and forget (yes, computers do forget);—and how computers work, timing everything in millionths of a second, and using a code of electric pulses that race at almost the speed of light.
- **different computer languages** (especially BASIC and LOGO)—including commands in these languages such as RUN, READ, SAVE, MOVE. (They sound familiar, don't they?)
- **different ways that computers affect our lives**—from "CAI" to "CAM," from robots, to bar codes, to data bases.

In addition, you'll find a pronunciation guide for each term, and a page with a pronunciation key to help you use each guide. You'll also find illustrations, both funny and wise, by an artist who sees computers with an "inner eye." Study these carefully, and you'll see with that eye, too.

What else could we do for you? Well, maybe, put this dictionary on a disk so you could read it on a computer. That wasn't possible this time. But to get you started, here's a "menu" for Scholastic's illustrated *Computer Dictionary:*

HI! WELCOME TO SCHOLASTIC'S ILLUSTRATED COMPUTER DICTIONARY. WHAT PART OF THIS BOOK WOULD YOU LIKE TO USE? SELECT THE NUMBER OF YOUR CHOICE.

# HOW TO USE THIS DICTIONARY

HI! YOU WANT TO KNOW HOW TO USE THIS DICTIONARY. HOW WOULD YOU LIKE TO START? SELECT THE NUMBER OF YOUR CHOICE FROM THIS MENU.

1. QUICK GUIDE ON HOW TO USE THIS DICTIONARY
2. HOW TO FIND A WORD
3. HOW TO READ A DEFINITION
4. WORDS TO LOOK UP FIRST

1. QUICK GUIDE ON HOW TO USE THIS DICTIONARY

• Use the *guide words* at the top of each page to find the page with the word you want. If you are looking for "GL," you'll find it between "gi-" and "go-."

• Track your word down by its first letter (*a, b, c,* etc.); then, by its second letter (*ca, ce, ch,* etc.); and so on.

• If there isn't a definition with the entry, follow the "see" reference to another entry.

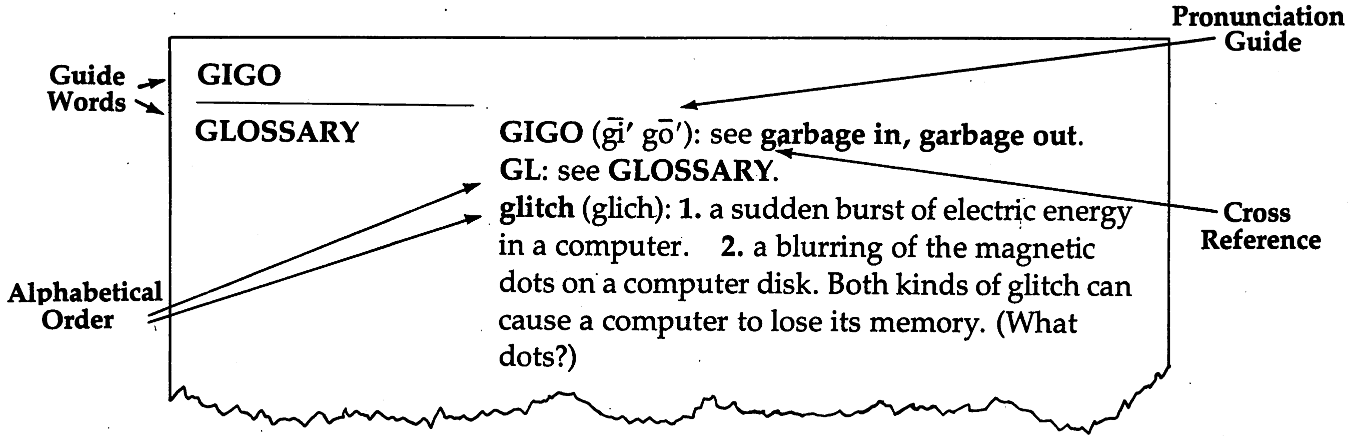
• Note the *pronunciation guide* with your word. If necessary, check it against the pronunciation key on page 6.

• If there are two or more definitions for your word (**1, 2, 3,** etc.), read each one. They may differ a whole lot.

RETURN TO THE TOP OF PAGE 4 IF YOU WANT TO SEE MENU AGAIN!

2. HOW TO FIND A WORD

Suppose one of your friends is upset because he lost a computer program in a "glitch." You decide to look up "glitch" in your *Computer Dictionary.* Entry words in a dictionary are arranged in alphabetical order. As you flip through the pages, you will see two *guide words* near the top corner of each page. The top guide word tells you the first word or term defined on that page. The bottom guide word tells you the last word or term defined on that page. When you come to the guide words **GIGO—GLOSSARY,** you know that "glitch" must be on this page. It's after GIGO and before GLOSSARY.



GIGO

GLOSSARY

GIGO (gī′ gō′): see **garbage in, garbage out.**

GL: see **GLOSSARY.**

**glitch** (glich): **1.** a sudden burst of electric energy in a computer. **2.** a blurring of the magnetic dots on a computer disk. Both kinds of glitch can cause a computer to lose its memory. (What dots?)

Sometimes, when you find the word you're looking for, there isn't a definition. A *cross reference,* with the word "see," tells you to look for another entry. (Usually, this happens when the entry is an abbreviation, or maybe a group of letters that stand for several words.) If you look up GIGO, for example, you find a cross reference to "garbage in, garbage out." Another tip: If the word you want begins with "un," and you do not find it, drop the "un" and look up the word that's left. You won't find UNPLOT, for example, but you will find PLOT. (UNPLOT is just the opposite.) Final tip: If you're looking up a term with two words (like "program instruction") and you don't find it, look up just the second word ("instruction").

RETURN TO THE TOP OF PAGE 4 IF YOU WANT TO SEE MENU AGAIN!

## 3. HOW TO READ A DEFINITION

After you've found a word, you need to know how to pronounce it. Following most entry words, you will see some letters and special symbols in parentheses. This is the *pronunciation guide.* To understand a pronunciation guide, you may want to refer back to the key on page 6. You will find instructions for "How to Use the Pronunciation Key" on the same page.

If you've flipped through this book, you may have spotted extra letters after some pronunciation guides. For example:

**input/output** (in′ pût oût′ pût), **I/O, IO.**

The extra letters are abbreviations, or short forms, for the entry.

A few more tips, as you start to read:

• Some entries include numbers in heavy type: **1, 2,** and **3.** The numbers show that there is more than one definition for the word.

• When you see parts of a definition marked (a), (b), (c), they indicate important points to remember.

• Does a definition begin with "in BASIC" or "in LOGO" or "in word processing"? These phrases are clues to a term's meaning. BASIC and LOGO are "languages" you use for talking with a computer. A *word processor* is a computer that lets you work with lots of correspondence, such as business letters.

• Does the definition include lines set off like this?

```
400 LET M = 45678
```

This is an example of an instruction you might put in a computer program. (This one is in BASIC.) Computers often need several instructions before they can show you results.

RETURN TO THE TOP OF PAGE 4 IF YOU WANT TO SEE MENU AGAIN!

## 5. WORDS TO LOOK UP FIRST

The following words are often used as part of other *Computer Dictionary* definitions. It might be helpful to look up these words before you start searching for other words:

| | |
|---|---|
| BASIC | data |
| bit | LOGO |
| command | program |
| computer | word processing |

RETURN TO THE TOP OF PAGE 4 IF YOU WANT TO SEE MENU AGAIN!

# HOW TO USE THE PRONUNCIATION KEY

Do you have a bookmark? Put it in this page. You'll probably be checking back here often. The key below will help you use the pronunciation guides you find with words in the *Computer Dictionary.* Each symbol in a word's pronunciation guide matches one symbol below. There are symbols for vowel sounds and consonant sounds. Notice the *key word* next to each symbol. The heavy letter gives you a clue to that symbol's sound. Here's how to use this key:

Take a look at the pronunciation guide for "abacus" on the opposite page. The key shows that the first sounds are: "a" as in bat, "b" as in bee. The next sound "ə" is the "schwa" sound, a sort of soft "uh." The last part begins with "k" as in key. There's that schwa sound again. The last sound is "s." Abacus, ab-uh-kuhs. Now get that bookmark!

**PRONUNCIATION KEY**

*VOWEL SOUNDS:*

| *symbol* | *key word* | *symbol* | *key word* | *symbol* | *key word* |
|---|---|---|---|---|---|
| a | **b**a**t** | i | bit | oi | **boil** |
| ā | bake | ī | bite | ou̇ | **out** |
| ä | **bar** | o | not | u | **nut** |
| e | bet | ō | note | ü | **rude** |
| ē | beet | ô | law | u̇ | **put** |

*There is another sound, ə, as in machine or over. It is called a "schwa" sound. It is not stressed, or accented. A schwa(ə) just sort of slips by you as "uh." (Uh?)

*CONSONANT SOUNDS*

| *symbol* | *key word* | *symbol* | *key word* | *symbol* | *key word* |
|---|---|---|---|---|---|
| b | bee | l | low | t | too |
| ch | **chew** | m | **me** | th | **thin** |
| d | do | n | no | t͟h | **then** |
| f | fee | n͟g | sing | v | vote |
| g | go | p | pie | w | way |
| h | how | r | ray | y | you |
| j | jaw | s | so | z | zoo |
| k | key | sh | **shoe** | zh | pleasure |

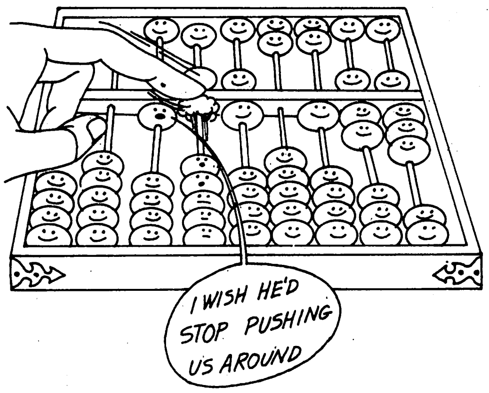
**Some More Clues**

- The pronunciation for each word is between parentheses, ( ). For example: **computer** (kəm pyü' tər).
- Each word is separated into syllables. (A syllable is the smallest chunk of sound you can break a word into.) For example, "program" has two syllables: **program** (prō' gram).
- When a word has more than one syllable, you "accent" one of them—you say it more strongly than the other(s). In "magnetic," accent the second syllable: **magnetic** (mag net' ik).
- Sometimes, you stress two syllables in a word. But one is lighter than the other. In "programmer," the light accent is on the second syllable: **programmer** (prō' gram'er).

# COMPUTER DICTIONARY

## A

**abacus** (ab′ ə kəs): a frame with strings of movable beads, used for counting. Each bead above the crossbar equals five beads on the same string below. The string to the far right can show the numbers 0 to 9. Strings to its left stand for tens, hundreds, thousands, etc. The abacus is set to zero when all the beads are pushed back from the crossbar. To count, you begin with the string on the far right. The Chinese abacus is the oldest computer known to humans — outside of our own brains. Students in Japan still use the abacus, even though their country makes calculators and other computers.



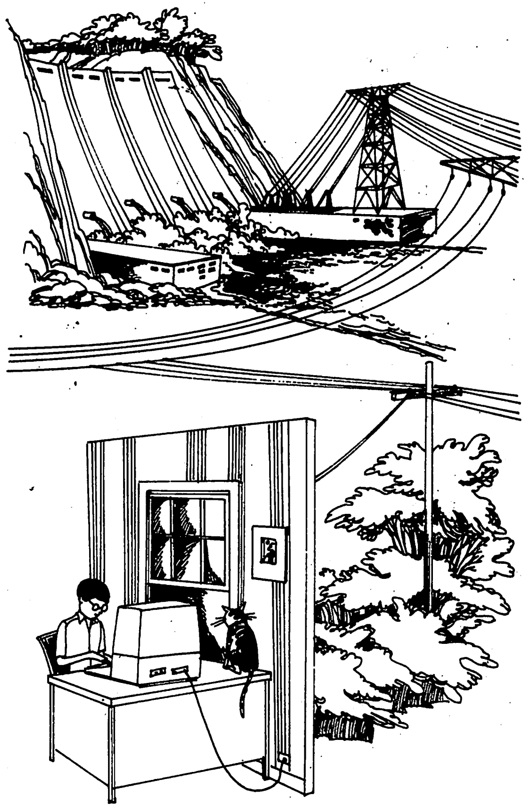
**abacus**

**ABS, ABS**olute value: a code in BASIC that tells the computer to remove the negative sign in front of a number. If T in your program stands for −5, ABS(T) would change it to 5. See **absolute value, negative number**.

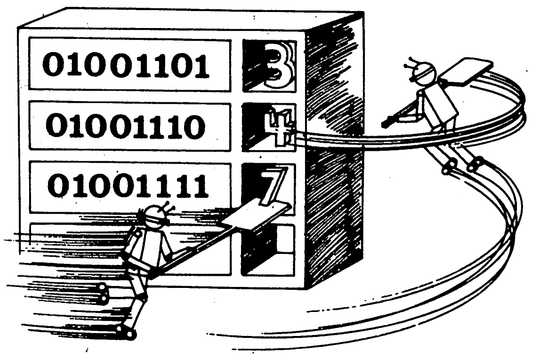
**absolute value** (ab′ səl lūt val′ yū): in mathematics, the amount that any number stands for, apart from its sign. The absolute value of 17 is 17. The absolute value of −17 is 17, also. See **ABS**.

**A.C.**, Alternating Current: the regular flow of electricity to your home and other buildings. Lamps light up with it. Computers plug into A.C. outlets, too!

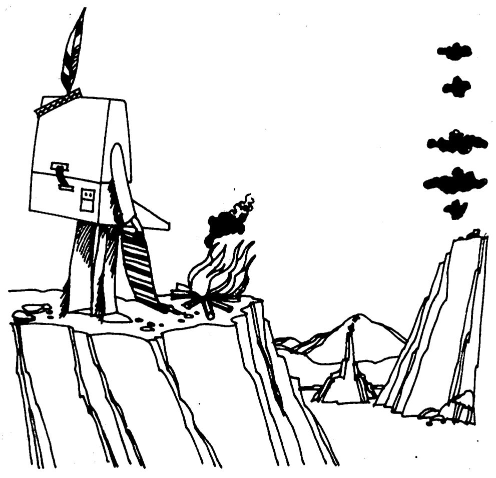
**access time** (ak′ ses tīm′): **1.** in general, the time it takes a computer to carry out one instruction. **2.** the time it takes to transfer a piece of data from a tape or disk to the computer's work area. **3.** the time it takes to get a disk or tape to start sending instructions into the computer's work area. A "floppy" disk gets started much faster than a tape. See **disk, tape**.



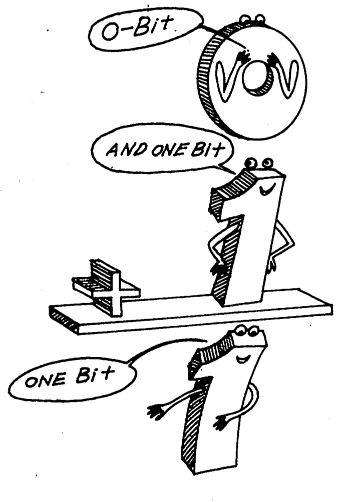
**A.C.**



accumulator



acknowledge



adder

**accumulator** (ə kyūm′ yə lā′ tər): a section of the computer's memory. The accumulator stores (a) numbers the computer wants to add, subtract, compare, etc., and (b) answers the computer gets from this work. The accumulator has many storage places, and each has a "number" name (for example, 01001101, 01001110, 01001111). If you tell the computer to add 3 and 4, it might
put 3 in accumulator space 01001101
put 4 in accumulator space 01001110
put the answer 7 in accumulator space 01001111
How does the computer actually add numbers? See **adder.** See **binary system**, too.

**ACK, ACKnowledge**: a coded message that one computer sends to another; a report that an earlier message got through without any mistake in it. See **NAK.**

**acknowledge** (ak nol′ ij): **1.** to tell another computer that its message arrived without any mistake. **2.** to press the ACK key, as a signal that a computer message was received. See **ACK, NAK.**

**acronym** (ak′ rə nim′): a word you make by taking letters from another word (or group of words). BASIC, the name of a computer language, is an acryonym. It is formed from the first letter in each of these words: Beginner's All-purpose Symbolic Instruction Code. Several entries in this dictionary are acronyms. See **mnemonic.**

**Ada** (ā′ də): a "high-level" computer language that was invented for American military use. It is used in science and industry, also. See **language.**

**adder** (ad′ ər): the place in a computer where numbers are added. They're not the numbers most people use (7, 8, 9, etc.). When you type your numbers into a computer, it changes them into its electric code. Each number becomes a string of tiny electric pulses ("bits") which travel to the adder. (A "one-bit" stands for a pulse that's "on." A "zero-bit" stands for a pulse that's "off.") The adder has four ways to combine the bits:

```
(a)  0      (b)  0      (c)  1      (d)  1
    +0          +1          +0          +1
    --          --          --         ----
     0           1           1         1, 0
```

The fourth answer (d) looks like 10 (ten), but it's not. In computer addition, it's "zero, carry one." The following strings of bits were added, starting from the right column:

```
 00111001  (first string of bits)
+10010110  (second string of bits)
 11001111  (answer: new string of bits)
```

See **binary system, arithmetic and logic unit, accumulator.**

**address** (ə dres'): the code number for a place where the computer stores a piece of information. When you type in a number, letter, or symbol, your computer (a) changes your letter into a string of electric pulses ("bits"), and (b) sends them off to a storage space. The "address" of this storage space is also a string of bits. So you might have 11001010 looking for its new home, 0100110000110101. Confusing? Not to worry: The computer post office (the control unit) uses other electric pulses to see that the letter gets to the right address! See **control unit.**

**ADP**, Automatic Data Processing: see **data processing.**

**AI:** see **artificial intelligence.**

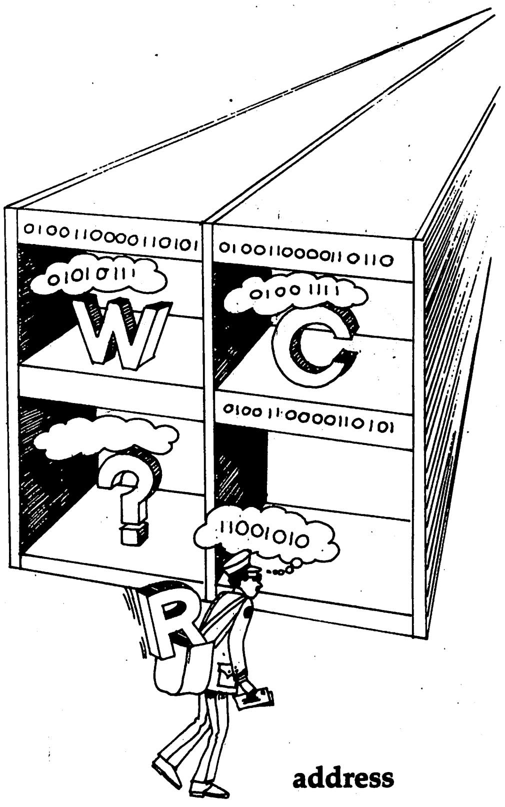
**ALGOL** (al' gōl), **ALGO**rithmic **L**anguage: a computer language used in mathematics and science. See **algorithm, language.**

**algorithm** (al' gə rith' əm): a rule, or set of steps, that can be used over and over to solve the same kind of problem. Algorithms save time in our work. Suppose you are in the business of laying carpets. To measure a room, you could block off each square foot, one by one, and then add all the blocks. Or, you could simply measure the length and the width of the whole room, and then multiply these two numbers. Your algorithm for this short cut is "area equals length times width." In BASIC, one way to tell the computer your algorithm is LET A = L*W. See **ALGOL.**

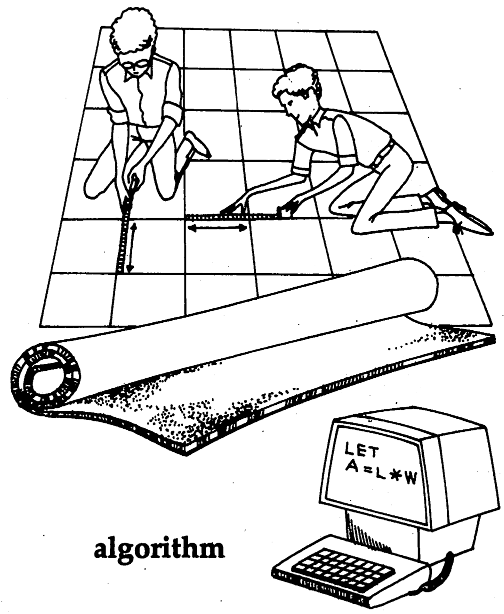
**alphanumeric** (al' fə nū mer' ik): made up of alphabet letters, numbers, and/or symbols. Most computer keyboards are alphanumeric. Here's an alphanumeric instruction in BASIC:

```
30 LET D = 5 + 8.
```

See **alphanumeric character.**



**address**

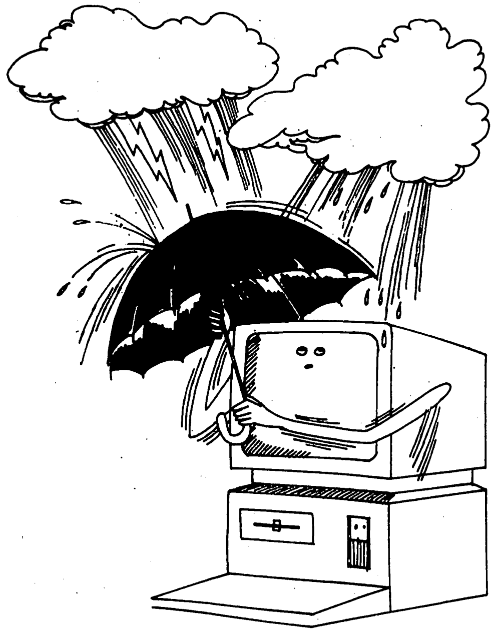


**algorithm**

**alphanumeric character** (al′ fə nū mer′ ik kar′ ik tər), **AN:** any letter of the alphabet (K, B, V), or number (7, 2), or symbol ($, *, ″). When you write a computer program, you use alphanumeric characters. Sometimes they're called "alphanumerics" or "alphamerics." See **alphanumeric, program.**

**ALU:** see **arithmetic and logic unit.**

**AN:** see **alphanumeric character.**



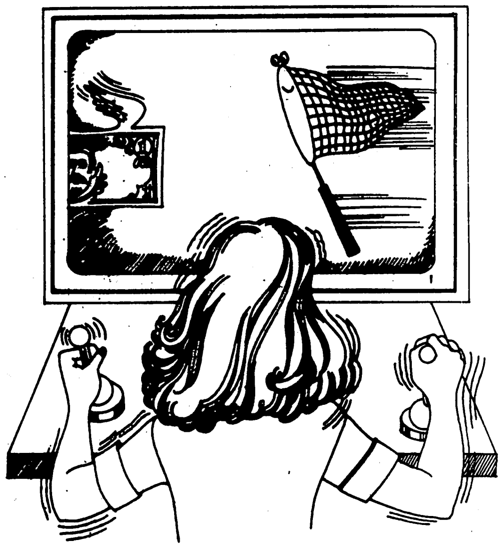
analog computer

**analog computer** (an′ əl ôg′ kəm pyū′ tər): a computer that measures a continuous change in physical conditions, such as temperature, light, air pressure, and so on. An analog computer has two tasks: (a) to measure conditions that are always changing; (b) to react with an electric signal that changes. Analog computers control instruments in automatic planes. In many ways, your brain is like an analog computer. It is always "reading" such things as temperature, air pressure, and the amount of oxygen in your muscles. Your brain responds to all this information by changing your heart rate. (That's why your heart pounds when you run fast, and why it slows down when you're not running anymore.) See **digital computer.**

**AND operator** (and′ op′ ə rā′ tər): an instruction that gets the computer to check its memory before taking a special step. If you want your computer to know when it should switch from one task to another, you can use an AND operator to set up "conditions." For example, you might instruct your computer that "if T equals 33 *and* if M is less than 100" it can jump to a new step. In BASIC, you might set up the instructions this way:

```
400 IF T = 33 and M < 100 THEN 420
410 GOTO 250
420 PRINT "NEXT JOB"
```

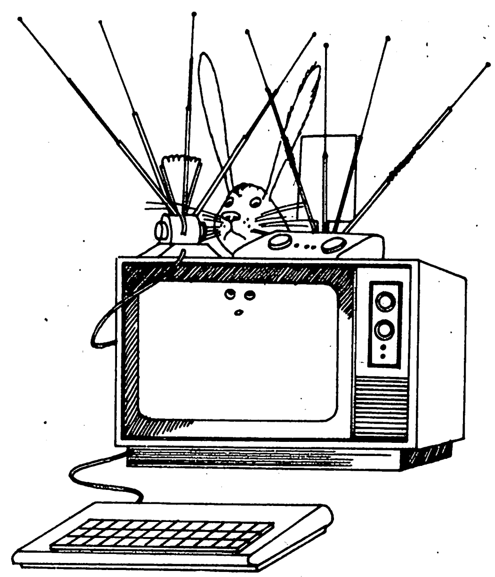
When it comes to the AND operator in line 400, your computer checks its memory. If it finds that both parts of line 400 are true, it will go forward to line 420 and print "Next Job." (If not, it will go to line 410.) Using the AND operator is like saying, "If I pass this test *and* if I do an extra assignment, I'll get a 'B' in English." See **logic operator, OR operator.**



animation

**animation** (an′ ə mā′ shən): the "movement" of an object across the computer screen, as in a video game. Actually, nothing moves. The computer just lights different parts of the screen very rapidly. Suppose you invent a game called "Disappearing Dollars." You want to show a dollar bill "flying" across the screen from left to right. You tell the computer how to shape the bill—maybe six spaces wide and two spaces deep—and to light up that much space on the screen. The next step begins the animation. You tell the computer to light up a new row of spaces along the right edge of the bill. At the same time you want it to lose a row of lighted spaces on the left side. By repeating these two steps very rapidly, you create the impression that the dollar bill is moving to the right.

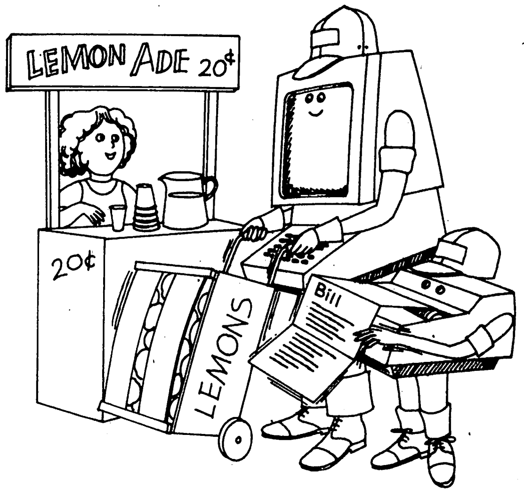
**antenna** (an ten′ ə): a rod or wire that attracts different kinds of "energy" signals, such as radio waves or TV signals. If you use your own television set as a computer screen, you may have to change your TV antenna. Your user's manual will tell you if this is necessary for your computer.


antenna

**APL, A Programming Language**: a computer language used especially in mathematics. See **language**.

**append** (ə pend′): to add at the end; to add a new record to the end of a computer file. See **file**.

**application program** (ap′ li kā′ shən prō′ gram): a pre-written set of instructions; a program that tells the computer how to do a specific job. Computers designed for small businesses usually "read" instructions in BASIC or COBOL (computer languages). A BASIC application program might tell such a computer how to make out workers' paychecks or how to bill customers. If it's a good program, hundreds of business owners may buy it from the "software" company that wrote it. Before buying an application program, it is important to know: (a) if the program is written in the language your computer knows; (b) if your computer has enough memory space to store the program. See **software, computer, dialect, BASIC, COBOL.**



application program

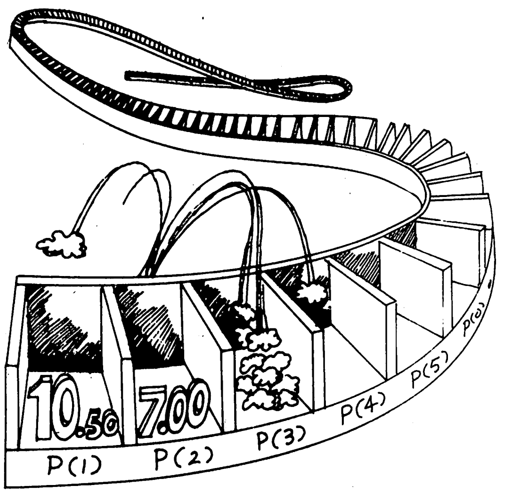
**arithmetic and logic unit** (ə rith′ mə tik ənd loj′ ik yū′ nit), **ALU**: the place in your computer where numbers are calculated and compared. The ALU is part of the computer's main work area—the central processing unit (CPU). See **computer, central processing unit, accumulator, adder.**

**arithmetic operator** (ə rith′ mə tik op′ ə rā′ tər): any symbol that tells the computer how to combine numbers. In BASIC, these operators are used: (+) for add; (−) for subtract; (*) for multiply; (/) for divide; (↑) or sometimes (**) for raising a number to a power of itself. For example, 3↑3 (or 3**3) would mean 3 times 3 times 3 (equal to 27). Whenever the computer finds numbers and an operator between parentheses ( ), it takes care of that arithmetic first. Can you see how the computer got the answer it produced for this instruction?

```
PRINT 4 + (3 * 5) − 5
14
```

See **logic operator, relational operator.**

**arithmetic symbol** (ə rith′ mə tik sim′ bəl): see **arithmetic operator.**



array 1.

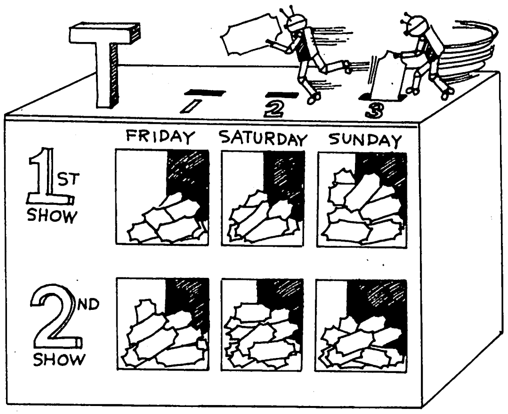
**array** (ə rā′): **1.** a place in your computer's memory where a list of data is stored. Suppose you want the computer to keep a record of your daily popcorn sales for a year. In BASIC, you can ask the computer to save a place called P(365) in its memory. If you wrote your array on paper, here is how it would start:

P(1) = $10.50
P(2) = $ 7.00

Your computer would save a row of 365 storage places—all under the code for P. As you typed each day's sales into it, the computer would store the information at one of these storage places. See **list, variable, subscript, DIM statement.** **2.** a place in your computer's memory where a table of information is stored. Suppose the owner of a movie theater wants to keep track of the tickets (T) he sells each day of a three-day weekend. There are two shows a day. He uses an array like this:

| | *Friday* | *Saturday* | *Sunday* |
|---|---|---|---|
| 1st show | T(1, 1) | T(1, 2) | T(1, 3) |
| 2nd show | T(2, 1) | T(2, 2) | T(2, 3) |

To set up this array, he tells the computer to save a place called T(2, 3) in its memory. (An array with two number places is sometimes called a *two-dimensional array.*) The 2 stands for the number of rows (shows) in his table; the 3, for the number of columns (days). The computer will store Saturday evening ticket sales in memory space T(2, 2), and so on. Arrays are powerful tools for keeping track of a lot of numbers. See **variable.**



array 2.

**arrow** (ar' ō): **1.** in BASIC, a symbol for multiplying a number by itself two or more times. (This is called *exponentiation.*) When you write 2↑3, you are telling the computer to multiply 2 times 2 times 2 (equaling 8). See **exponentiation.** **2.** in program planning, the symbol you use in a flowchart to point from one step to the next. See **flowchart.** **3.** in some computers, a key that you touch to erase the letter you just typed. In other computers, however, this key does just the opposite! It can be used to switch from where you are working to another part of the screen. In this case, the arrow moves the "cursor" (a tiny light that tells you where the next letter or number will appear), but it doesn't erase anything. (You have to check your computer manual on this one!) See **cursor.**

arrow

**artificial intelligence** (är' tə fish' əl in tel' ə jəns), **AI: 1.** a computer's ability to carry out tasks that our brains can do. Humans and computers do arithmetic, compare numbers, and repeat steps in 1-2-3 order. Humans and computers can pick the best way to do a task. But only humans can change their minds. That's natural. **2.** a study, or "model," of how people learn. Computer scientists believe it is possible to build an artificial brain (like an artificial heart). See **robot, bionics.**

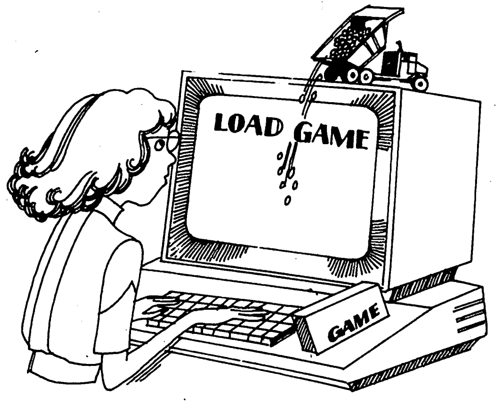
**ASCII** (as' kē), American Standard Code for Information Interchange: a computer code that turns letters, numbers, and symbols into electric pulses ("bits"). For example, W is represented by pulses that go like this: off, on, off, on, off, on, on, on. There's a short way of writing that! Use "1" for "on," and "0" for "off." Then the ASCII code for W looks like this: 01010111. Here's WOW in ASCII:

| | |
|---|---|
| W | 01010111 |
| O | 01001111 |
| W | 01010111 |

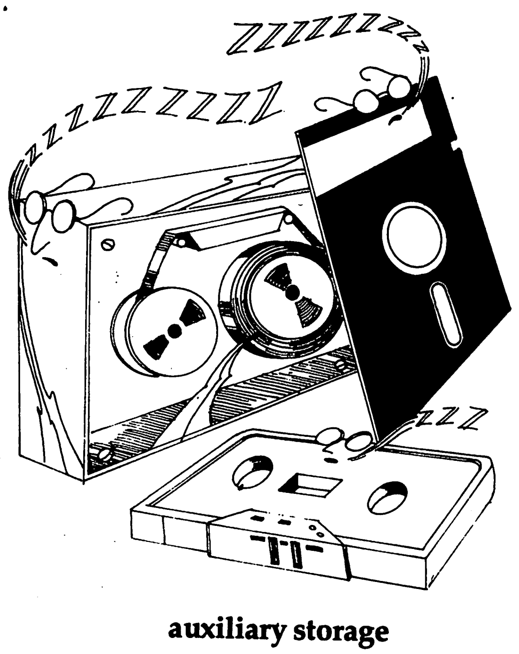
ASCII is a standard code. It is used by different computers that can talk to one another. See **language.**

**artificial intelligence**

**assembly language** (ə sem' blē lang' gwij): a computer language that uses letters — but not words — to "talk" to the computer. An assembly language is a "low-level" language. That is, the computer "reads" it very easily and does not use up much storage space (or time) to work out problems with it. See **language.**



assignment



auxiliary storage

**assignment** (ə sīn′ mənt): **1.** a number or value you give the computer to work with. In BASIC, you "assign" the number 30 to your computer by typing LET A = 30. See **LET.** **2.** a message to the computer, telling it to use information on a disk or tape. If you have stored a computer game on a cartridge, you could, in BASIC, "assign" the computer to LOAD GAME. Your game would be ready to play, in seconds. (A nice assignment.)

**authoring language** (ô′ thər ing lang′ gwij): any computer language that would help you to write programs while you sit in front of the screen. These languages are just being developed, but in a few years, most people will use authoring languages for everyday needs. Just imagine coming home from work, going over to your computer, and seeing this on its screen:

```
HI! DO YOU WANT TO:
    INVENT A NEW GAME?
    WRITE A CHECKBOOK PROGRAM?
```

Sure. See **language.**

**AUTO** (ô′ tō): in BASIC, your command to the computer to number each line in the program you are typing. With AUTO, the computer will number your lines by jumps of ten (10, 20, 30, etc.). You can start the command at any number. With the command AUTO 410, for example, the computer would call the first line 410, the second line 420, and so on. See **line number.**

**auto-load** (ô′ tō lōd′): on some computers, a button or key that you push to make your computer active or to get it ready to work for you. See **operating system.**

**automatic data processing** (ô′ tə mat′ ik dā′ tə pros′ es′ ing), **ADP:** see **data processing.**

**autoscore** (ô′ tō skôr′): in word processing, an instruction for your computer to underline what you type. After you put your computer in autoscore, it will keep underlining until you cancel that instruction.

**auxiliary memory** (ôg zil′ yər ē mem′ ə rē): see **auxiliary storage.**

**auxiliary storage** (ôg zil′ yər ē stôr′ ij): a place for storing information that the computer does not need at the present time. Auxiliary storage includes cartridge tapes and plastic disks. See **storage, disk.**

## B

**Babbage, Charles** (bab′ ij chärlz′), 1791-1871: an English mathematician who designed a computing machine in 1833. Babbage hoped his "Analytical Engine" would solve the kinds of problems that we give to modern computers. But it was never built. There weren't any tools for making it! Babbage died broken-hearted.

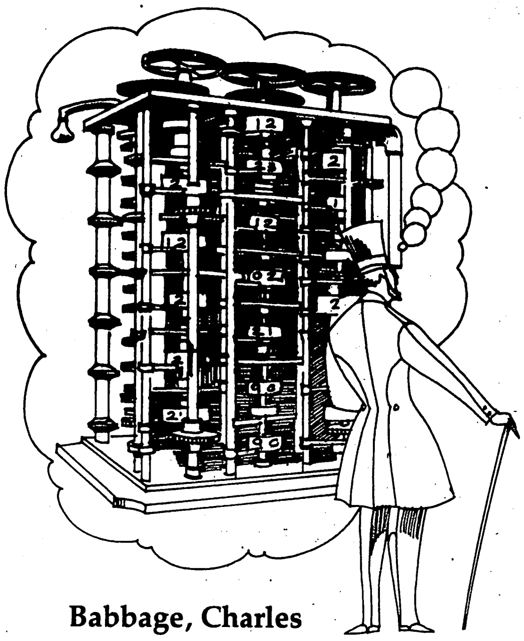
**BACK** (bak), **BK**: in LOGO, a command that moves a small figure (a "turtle") on the computer screen. The command must be followed by a number (for example, BACK 15). When you press the RETURN key, the turtle will back up that many steps. See **FORWARD, turtle, LOGO.**

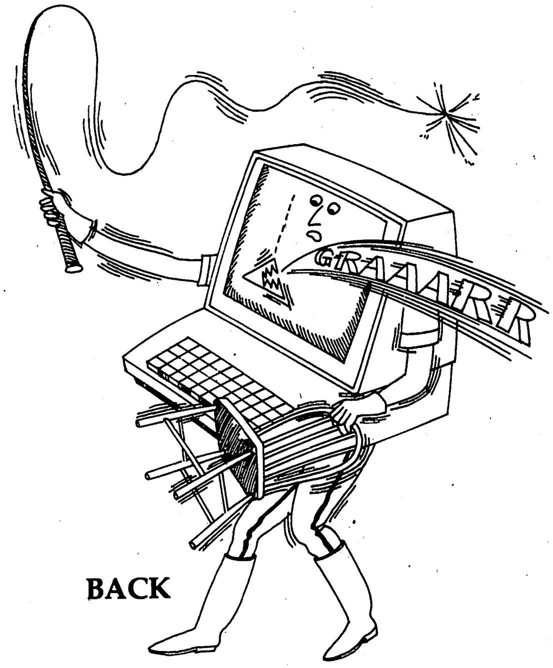
**backspace** (bak′ spās′): **1.** to go back one keystroke on the line you are typing into the computer. **2.** to reverse a magnetic tape to the beginning of a message. See **BACKSPACE, cursor.**

**BACKSPACE, BS:** in word processing, the command that moves the cursor (a small light on the screen) one place to the left. Backspacing allows you to change what you already typed. See **cursor.**

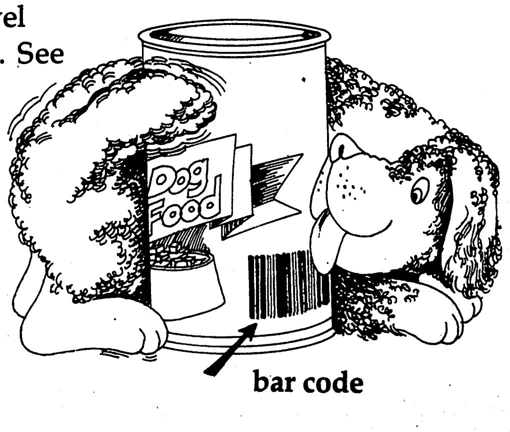
**bar code** (bär kōd): a pattern of stripes ("bars") that you see on book covers and other store items. The code stands for the brand-name, type, and size of each product. Bar codes can be "read" at the check-out counter by a "wand"—an object like a thick crayon—or other scanner. These sense the width of the stripes in a bar code. They send this information to the store computer which identifies the item and tells the cash register to print the price. All you have to do is pay. See **wand, optical character recognition.**

**BASIC** (bā′ sik), **Beginner's All-purpose Symbolic Instruction Code:** a "high-level" computer language that lets you "hold a conversation" with your computer. BASIC was invented at Dartmouth College. It is made up of English words, abbreviations, and symbols from mathematics. BASIC is the most popular high-level language for minicomputers and microcomputers. See **language, minicomputer, microcomputer.**

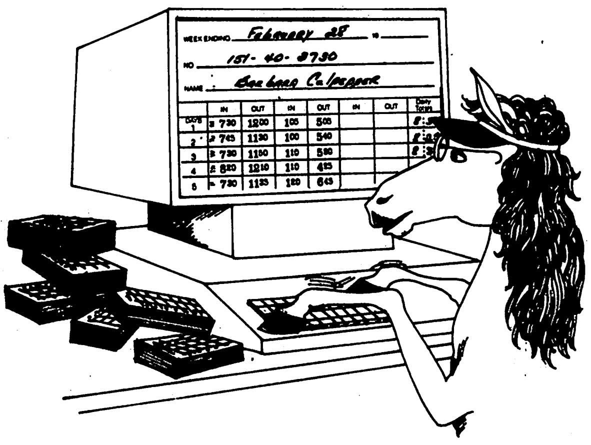

**Babbage, Charles**



**BACK**



**bar code**



batch processing

**batch processing** (bach′ pros′ es′ ing): a way to schedule jobs for a computer that works with large amounts of data. Suppose the Horseshoe Company has thousands of employees. It will probably use batch processing to make out its workers' paychecks. The names and salary rates of Horseshoe's employees are always in the computer system. So is the program that tells the computer how to make out their paychecks. But the payroll program will run only once every week or two, after a "batch" of many employee timecards has been collected. Batch processing won't work with every kind of job. Airline computers, for example, make each traveler's reservation immediately instead of waiting for a batch of many requests.

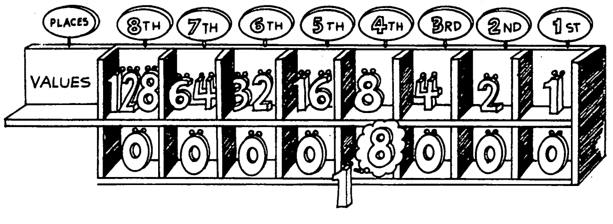
**Beginner's All-purpose Symbolic Instruction Code** (bi gin′ ərz ôl′ pər′ pəs sim bol′ ik in struk′ shən kōd′): see **BASIC**.

**beginning-of-tape** (bi gin′ ing uv tāp′), **BOT**: a marker that shows where to start recording on a computer tape; sometimes a transparent section of the tape.

**BEL, BEL1**: code for a warning sound that goes off when the computer notices an error. Usually, the mistake is in data arriving from another computer location. The BEL code is used in middle-size and large computers.

**binary** (bī′ nə rē): having two different parts. In mathematics, the binary system has only two numbers, 0 and 1. Another system, the decimal system, is more familiar to most people. But every computer talks in some combination of zeros and ones. See **binary system, digital computer**.

**binary digit** (bī′ nə rē dij′ it): see **bit**.



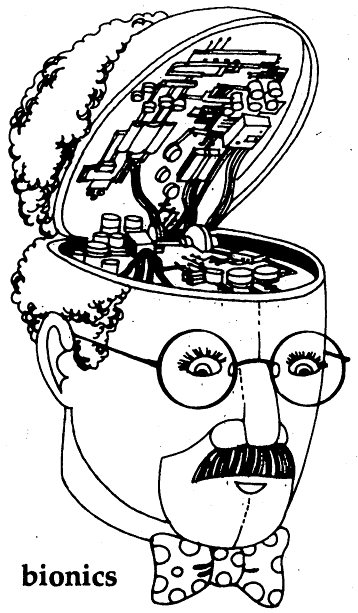
binary system

**binary system** (bī′ nə rē sis′ təm): a counting system that uses only two numbers, 0 and 1. The zero is just a place-holder. The "one" equals whatever place it's standing in. Below are the values of the first eight places in binary. (The first place is equal to 1, the third place is equal to 4, and so on.)

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| *places:* | (8) | (7) | (6) | (5) | (4) | (3) | (2) | (1) |
| *values:* | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |

Beginning with 2, the value of each place is twice as large as the place to its right. Suppose you want to write the binary number that equals 8. You place a "1" in the fourth place, and put 0s in all the others: 00001000. To write 12 in binary, put a "1" in the fourth place, a "1" in the third place, and zeros in the others: 00001100. If you put a "1" in every place, you have the value of 255. (Add all the values.) These eight places have a special use. They match the eight places in most computer "bytes" (computer code). Computers love the binary system. It fits right in with the only two things a computer "brain" can do: pulse on and off. See **byte, language, adder**.

**bionics** (bī on′ iks): a science that compares how humans and machines are able to work. Humans regulate their body heat; thermostats turn furnaces on and off. Humans can add and compare numbers; computers are designed to do the same thing. Can we build a machine that is fully human? Not unless there's a machine that can build us. See **robot, artificial intelligence**.


bionics

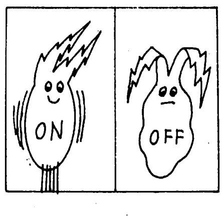
**bit** (bit), **binary digit: 1.** in print, a "1" or a "0"; a one-bit or a zero-bit. A bit is the smallest unit of code that computers read. Usually bits are in groups of eight. Each group stands for a letter, number, or other symbol that humans use. See **ASCII. 2.** in a working computer, a pulse of electricity ("on"), or a pause between two pulses ("off"). Computers work by electricity—by tiny pulses that flow at controlled rates. When you type the letter L, the L-key touches off a string of these electrical pulses and pauses. They flow to a central control area, to a work area, to a storage area, and back to the screen where they produce the same letter again. It all happens very rapidly. Some computers can carry out a half-million instructions in the time it takes to type "L." See **binary, byte, nibble, word**.

**bit/s**: see **bits per second**.

**bits per second** (bits pər sek′ ənd), **bit/s, BPS**: the rate at which data travels between a computer and some outside connection. This rate is measured in the number of electric pulses that pass one point in a second. See **bit**.

**BK**: see **BACK**.

**BL, BLank**: computer code for an empty space in a message that is being sent to another computer. The code is included so that the computer receiving the message does not think something is missing. Also, NUL.

bit


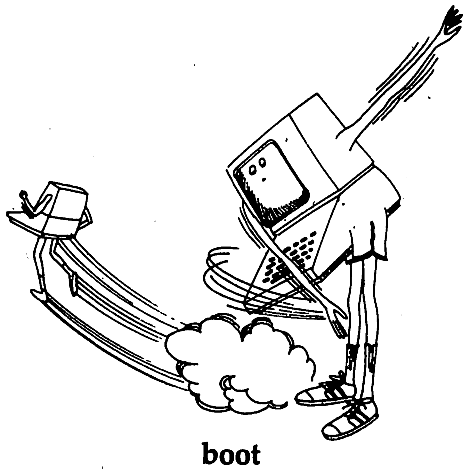
**blank PRINT** (blangk print): in BASIC, a way of using the PRINT command to put an extra space between two lines. If your computer has a printer, the printer will move line by line down a page, unless you tell it not to. If you want it to skip a line someplace, you include that instruction right in the middle of your program: You type the word PRINT, but leave the rest of the line blank. Suppose you want to skip a line between the title and the first line of your poem. This is how you set it up:

```
300 PRINT "SOME BITS"
310 PRINT
320 PRINT "SMALL PULSES RACE . . ."
```

Here's what you see when your computer "runs" this:

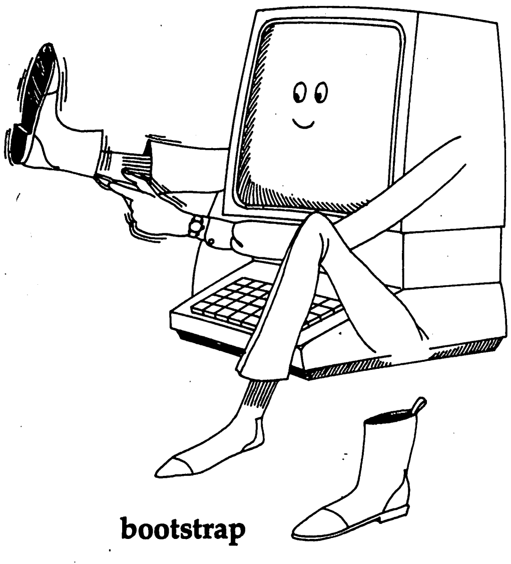
```
SOME BITS

SMALL PULSES RACE . . .
```

(Want to finish the poem?) See **PRINT, printer, program.**


boot

**boot** (būt): to get a computer ready to work; to load all the instructions a computer needs before it starts working on your program. It's like doing warm-ups before a race. You get your whole system going! See **bootstrap**.

**bootstrap** (būt' strap'): what your computer needs before starting its workday. You help it take the first step by turning a switch. This wakens the computer's "permanent" memory, which remembers to look for the set of instructions it always follows when it starts. It finds them, and these instructions tell it to go looking for others! By now, the computer is reading instructions from its "operating system." Within a second or two, it is all ready for the day's work. Why is this process called a "bootstrap"? Some boots have strips of leather sewn above the heel. When you want to put on these boots, you pull them up by the straps. (There's an old saying: "I pulled myself up in life by my own bootstraps.") It's as though the computer uses the bootstrap to get itself awake. (But don't forget: *you* pull the first switch!) See **boot, operating system, auto-load.**


bootstrap

**BOT**: see **beginning-of-tape.**

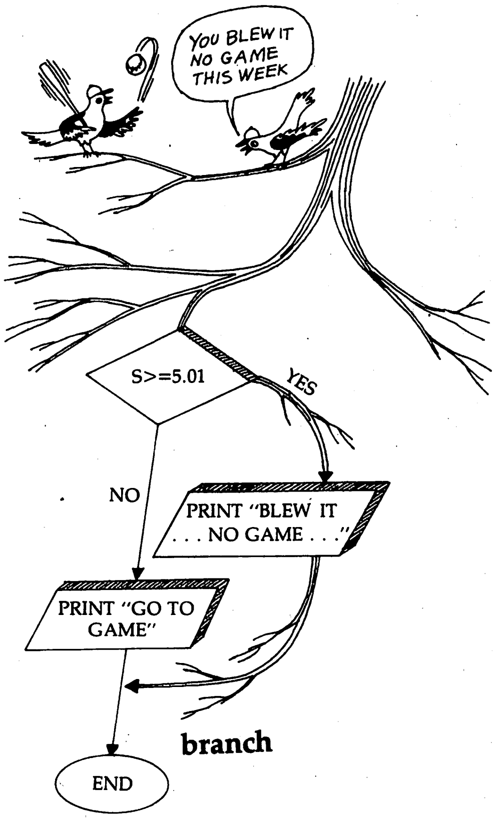
**BPS**: see **bits per second.**

**brain** (brān): an item the computer doesn't have. But you'll often hear that the brain of a computer is its "processor." That's where it does brainy things like adding and comparing. See **central processing unit, artificial intelligence, bionics, robot.**

**branch** (branch): **1.** a place in the outline of your program (your "flowchart") that shows where your computer must make a decision about its next step. See **flowchart, decision box.** **2.** a program instruction that tells your computer to decide whether it should skip the next step. Suppose you write a program to keep track of your allowance. If the amount you spend (S) is more than $5.00, you can't go to the softball game this weekend. Here's the part of your program (in BASIC) where you include a branch:

```
500 IF S > = 5.01 THEN 530
510 PRINT "YOU CAN GO TO THE GAME!"
520 GOTO 599
530 PRINT "YOU BLEW IT! NO GAME!"
599 END
```

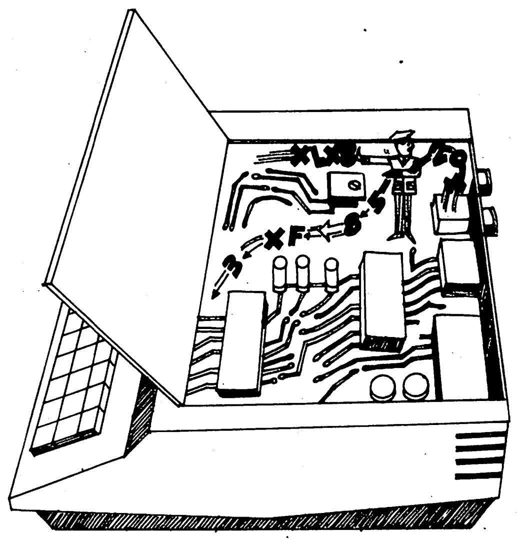
Line 500 sets up a **conditional branch:** The computer will jump to line 530 if you spent over your $5.00 limit. If you stay within your limit, the computer will go on to line 510 (then 520, then 599). On line 520, the computer *must* skip a step — no ifs about it. It's an **unconditional branch.**

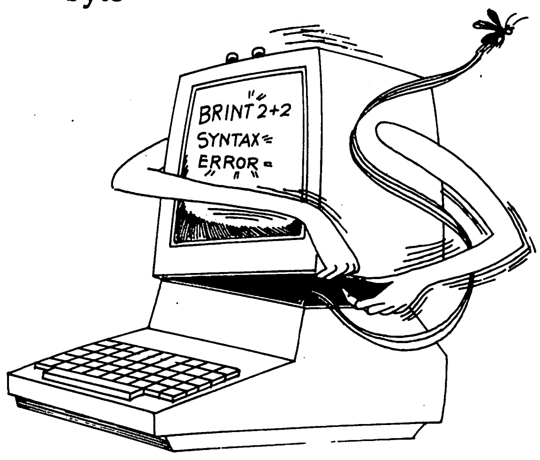

YOU BLEW IT NO GAME THIS WEEK
S>=5.01
YES
NO
PRINT "BLEW IT . . . NO GAME . . ."
PRINT "GO TO GAME"
END
**branch**

**BREAK** (brāk): on some computer keyboards, a key that you touch to stop the action of the computer. With the BREAK key, you can (a) interrupt a program that the computer is running, or (b) interrupt the transfer of data between a tape and the computer's active memory.

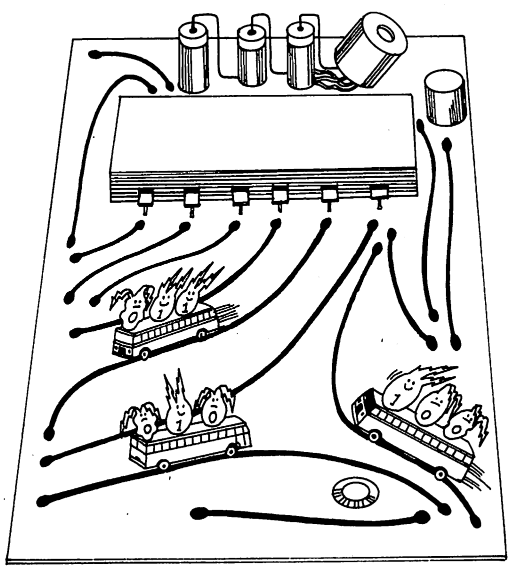
**BS:** see **BACKSPACE.**

**buffer** (buf' ər): any place in a computer system where the flow of data is slowed down or speeded up on purpose. Data travels much faster in some parts of the computer system than others. In the computer's central work area, data moves with lightning speed; but in a tape recorder it travels much more slowly. So when you tell the computer to store a program on tape, it sends your program through a buffer. The buffer is like a line outside a football stadium. A thousand people may arrive at the same time, but they all can't go through the gate at once. They have to wait — sometimes for quite a few minutes. A piece of computer data, however, waits in a buffer just a few seconds.


**buffer**



bug



bus

byte

**bug** (bug): **1.** a flaw in the design of a computer. People who build computers spend a lot of time testing them for "bugs," trying out all the possible jobs their computer might be asked to do. See **debugging.** **2.** a mistake you might make in writing a computer instruction. If you misspell a command, you will stump most computers (although some computers do know how to "guess"). Suppose you want to type the BASIC command, PRINT, but you type PRINN, instead. Your computer will probably flash SYNTAX ERROR as a way of saying, "Mistake?" See **test, SYNTAX ERROR.**

**bus** (bus): a major path for transferring data within a computer. It is a path that electric pulses ("bits") follow, as they move to the storage area, the arithmetic work area, and other places in the computer's system. A computer bus is something like our major highways, but not like the trucks and buses you see on them!

**BYE** (bī): in BASIC, a message to the computer that you have no more work for it right now. You use BYE if your computer keyboard is connected to a computer center someplace else. You don't have to say BYE to your own computer (unless you really want to). See **log off, time sharing.**

**byte** (bīt): **1.** in a working computer, a string of electric pulses ("ons" and "offs") that the computer "reads" as a piece of code. Usually, a byte includes eight bits. **2.** in print, a string of ones and zeros (1s and 0s) in a piece of computer code. Usually, there are eight bits in a byte, but a byte can be shorter or longer. From 00000000 to 11111111, there are 256 ways to combine 1s and 0s. (Try it!) Each of these 256 bytes can stand for a letter, number, or other symbol that we use in human language. The fun starts when you put two bytes together. There are 65,536 ways to do that! And that's the secret of a computer's memory! It now has thousands of ways to code your work. Many personal ("desktop") computers have 65,536 (64K) bytes of memory. See **bit, nibble, word, binary.**

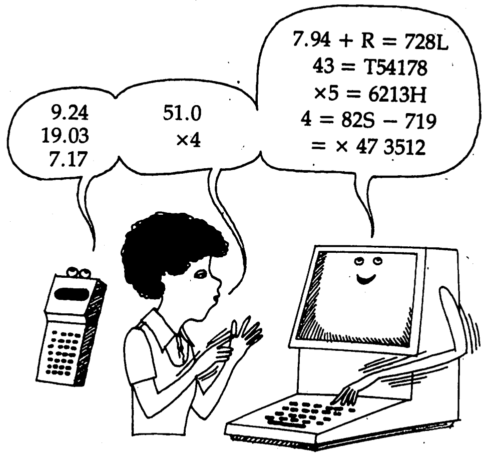
# C

**CAD:** see **computer-aided design.**

**CAI:** see **computer-aided instruction.**

**CAL:** see **computer-aided learning.**

**calculator** (kal′ kyə lā′ tər): any person or object that solves problems by using mathematical rules. Small "pocket" calculators are popular in math classes. You're a calculator every time you count change! Most computers are calculators, too, but they can do much more.



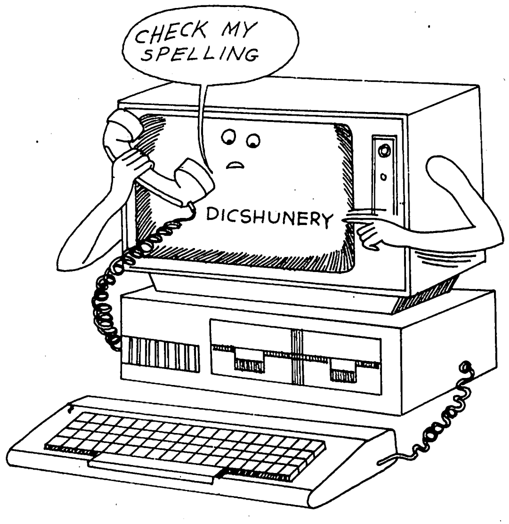
**calculator**

**call** (kôl): **1.** for a computer, to leave the program it is working on, in order to find instructions it learned some other time. Most computers have certain "routines" they can always turn to, such as finding the square root of a number. If you give the computer a new program that includes square roots, here's what your computer does: It (a) "calls" the square root instructions in its memory; (b) turns its work area over to these instructions; (c) leaves directions for getting back to your program after the square roots are figured. **2.** for a computer, to repeat steps it did earlier in the same program. If a computer knows it has to repeat lines 340 through 390 in your program, it stores them once. Then, each time it needs them again, it "calls" them back with its code for line 340.

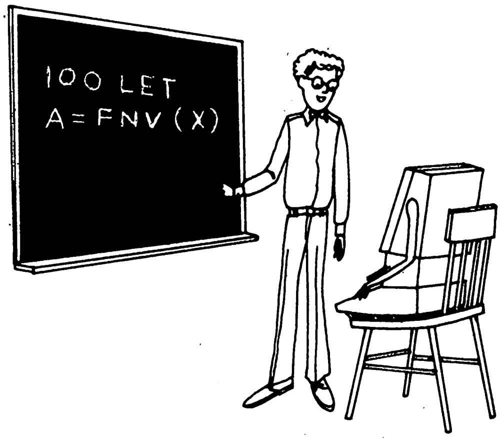
**CALL:** in BASIC, your command to the computer to find a set of instructions already in its memory. To use CALL, you have to know the exact location of data in your computer's memory. Your computer manual may list the storage numbers in your computer's memory. Suppose it does, and you know that the first instruction for finding the square root is in the storage area "900." On line 50 of your program you might type:

```
50 CALL 900
```

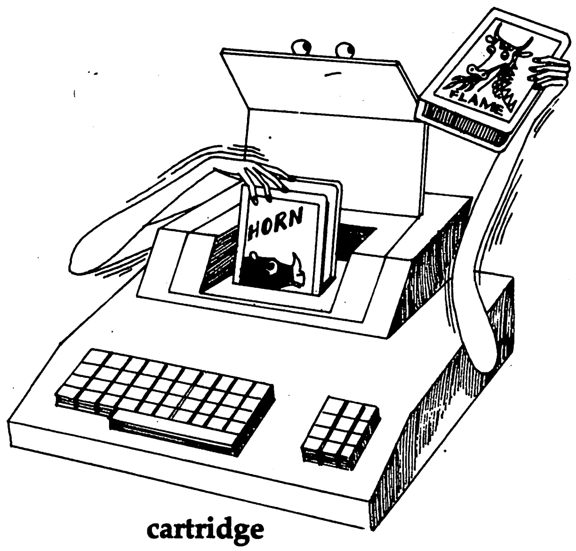
Your computer would go directly to this storage area and follow the instruction there. Then it would go to 901, 902, and so on, until it finished all the steps for finding square roots. When you use these addresses, you are writing in "machine language." See **POKE** for why you might want to learn some machine language.



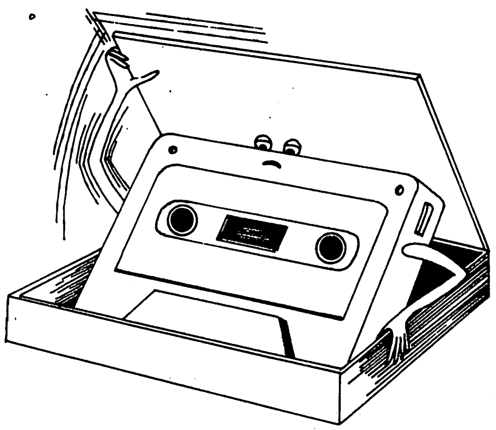
**call**



calling statement



cartridge



cassette

**calling statement** (kô′ ling stāt′ mənt): your direction to the computer to find and use instructions stored in its memory. For BASIC users there are three types of calling statements. First, you can ask your computer to find and use a rule its designer taught it—a "library function." Your computer may already have a rule for finding square roots, for example. Here's how you might ask the computer to use it to find the square root of 11236.

```
PRINT SQR (11236)
```

Then there are the rules you teach your computer and tell it to store away — the "user-defined functions." Suppose you taught your computer a function (FN) to predict the cost of a vacation trip (V). This could be your calling statement.

```
100 LET A = FNV(X)
```

And finally, there are instructions called a "subroutine" you might want the computer to repeat in the same program. If these instructions begin on line 200, this could be your calling statement.

```
700 GOSUB 200
```

Calling statements save the computer time and space because each instruction can be stored just once. See **library function**, **user-defined function**, **subroutine**, **CALL**.

**CAM**: see **computer-aided manufacturing**.

**CAN, CAN**cel: code telling a computer to ignore the word, line, or message block that was just sent by another computer. See **CANCEL**.

**CANCEL** (kan′ səl): in word processing, a key that you touch when you change your mind about an instruction. For example, suppose you want to get rid of a line in a poem you are writing on the computer. You touch DELETE, a key you can use to erase a line. But then you decide not to take out the line. You touch the CANCEL key, and your poem is still the same. (You "canceled" DELETE.) See **CAN**.

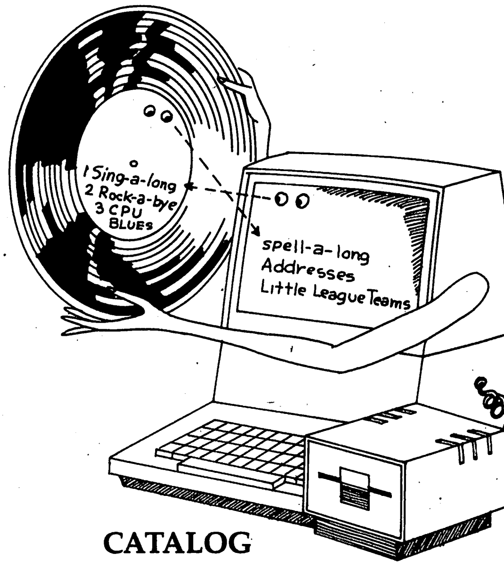
**CAPS** (kaps): see **upper and lower case**.

**carriage return** (kar′ ij ri turn′), **CR**: see **RETURN**.

**cartridge** (kär′ trij): a removable plastic case that holds a magnetic tape or disk and is used to store computer programs. Cartridges are easy to insert either into your keyboard terminal or into a separate "drive" (a player). Some cartridges (for example, in the Atari 800 system) hold permanent programs to which you can add data. You can't change the instructions inside one of these cartridges, but you can supply your own facts when you use it. See **disk, disk drive, cassette.**

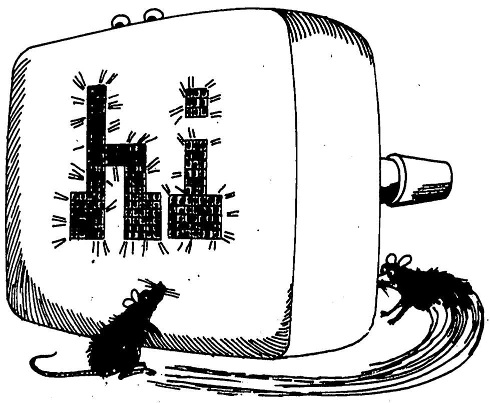
**cassette** (kə set'): usually, a small, removable case with a tape inside for recording computer data. Cassettes are the least expensive way to store programs that you write and want to save. With most computers you can use your own tapes and cassette recorder. It's the easiest way to share computer programs with friends!

**CATALOG** (kat' ə lôg'): in LOGO and in BASIC, a command to your computer to name all the files being held on a magnetic disk. Using CATALOG is like looking at the label on an LP record. The LP label will tell you the names of the songs you can hear. CATALOG gets your computer to tell you the names of files you can use. Your files might include programs you have invented, or anything else that interests you. See **disk, file.**



CATALOG

**cathode-ray tube** (ka<u>th</u>' ōd rā' tūb'), **CRT**: a screen for your computer, similar to the one on your TV. The computer CRT shows (a) what you type into a computer and (b) results from the computer's work. The CRT is a vacuum tube. When it's on, tiny charges of its electricity race toward a phosphor screen. Your computer controls the exact path of each stream of electricity. These streams, or "rays," hit your screen in the shape of the letter or number you type. Phosphors light up when they are excited. So whenever you type "Hi" on your keyboard, you excite a lot of little phosphors. See **visual display unit.**

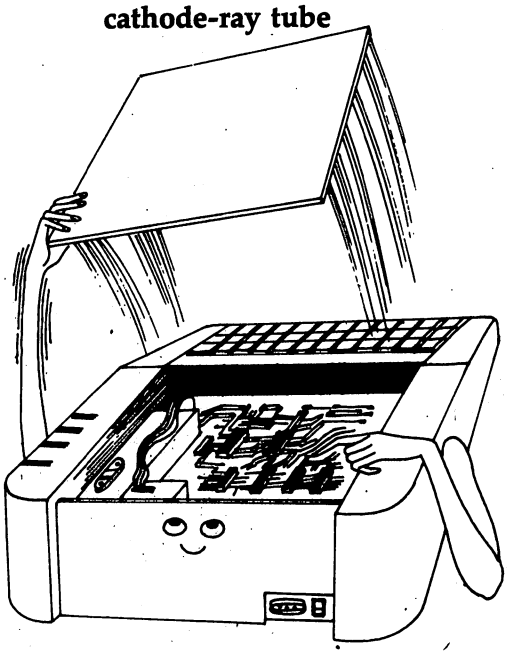


cathode-ray tube

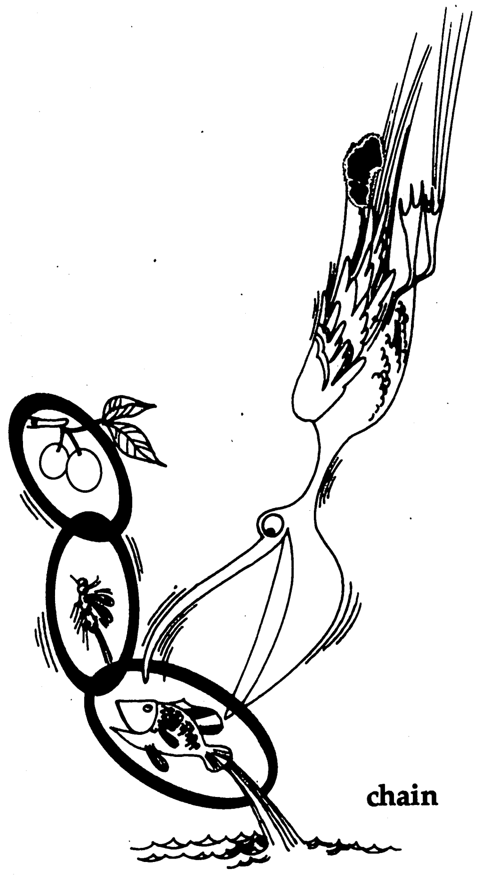
**CENTER** (sen' tər): in word processing, a command telling the computer to print a line so that it is the same distance away from the margin on each side. Within this definition,

this line is centered!

**central processing unit** (sen' trəl pros' es' <u>ing</u> yū' nit), **CPU**: where all the action is, in a computer! The CPU receives every instruction coming into the computer and sends out every result. It has (a) a control unit, (b) work areas including the arithmetic unit, and (c) a place to store results. Everything works by a clock that measures time in millionths of a second. The CPU is sometimes called the "heart" of the computer, or its "brains." See **control unit, arithmetic and logic unit, circuit, clock, storage.** (And see **brain!**)

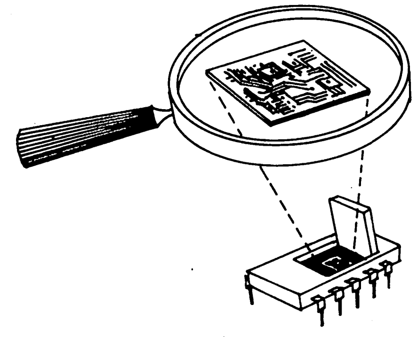

central processing unit

**chain** (chān): **1.** a series of steps in which the answer to one step becomes part of the next step. The following will be a chain if you add each number as you come to it: 2 + 3 + 1 + 4 + 2 + 1. **2.** in the computer's memory, a set of items that are related, but not always next to one another. Such items form a "chain" because each one contains the instructions for finding the next one. They're something like the cross-references in this dictionary. Sometimes an entry has a "See" reference at the end that points to the next definition you can check. Together, they're a "chain" of clues.


chain

**character** (kar′ ik tər): **1.** a letter, number, or symbol that always means the same thing. Some symbols such as ? or $ are easy to decode. See **alphanumeric character.** **2.** any pattern of electric pulses ("bits") that has a standard meaning among computer designers. For example, the bits 0000110 are international code for ACK ("message arrived OK"). See **ASCII.** **3.** sometimes, the amount of storage needed to hold a character (as defined in 2 above).

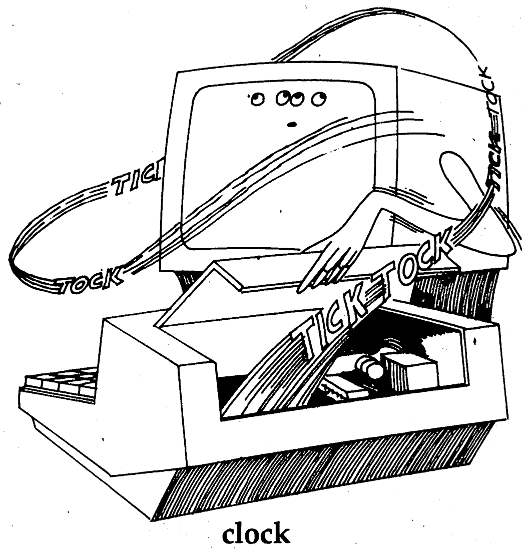
**chip** (chip): the workhorse of your computer; a tiny piece of silicon (made from sand) inside your computer. How tiny is a chip? The head of a typewriter key is larger. A chip is covered with very fine paths called "circuits." (In fact, a chip is sometimes called an *integrated circuit*.) A chip works on a tiny current of electricity that moves along these circuits. Chips do different kinds of work, depending on how their circuits are designed. In a small computer, one chip may hold enough circuits to do all of that computer's work. Larger computers have several different chips: one chip to add and subtract; another chip to work as "memory," and so on. Chips are sealed in small cases to protect them from dust. One speck, and your computer would pop corn. See **semiconductor, circuit, transistor.** And see **central processing unit.**


chip

**circuit** (sûr′ kit): a path that electricity follows. There are thousands of circuits in a computer. Electric pulses ("bits") race along circuits to add, compare numbers, store answers, and do other tasks. Circuits on a computer chip are smaller than the tiniest veins in a flower petal. But computer circuits do not grow naturally. They are designed by people using other computers. The designs are then burned chemically onto a thin wafer of silicon. (Silicon is an element found in sand). See **semiconductor, chip, transistor, silicon.**

**clear** (klēr): to remove pictures, numbers, words, or other content from the computer screen.

**clock** (klok): an electric signal inside the computer that controls its work time. Some computer clocks divide a second into a billion parts. The computer designer decides how many billionths to allow for one step in a program. Every instruction is then carried out at the same rate: step-step-step-step.



clock

**CMI**: see **computer-managed instruction**.

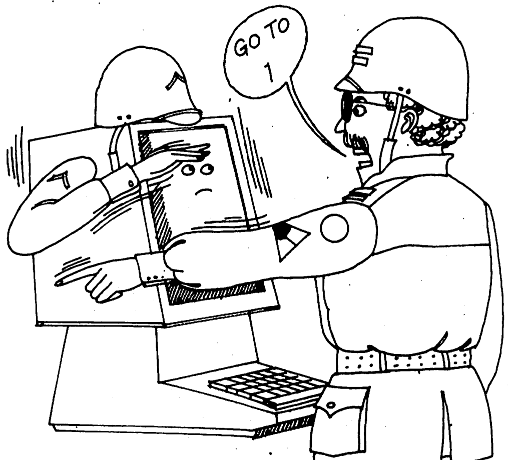
**COBOL** (kō' bôl'), **CO**mmon **B**usiness-**O**riented **L**anguage: a "high-level" computer language for people in business. See **language**.

**code** (kōd): **1.** for the code that computers "talk," see **ASCII**. **2.** for the code that opens up the secrets of some computers, see **personal code, password**.

**command** (kə mand'): **1.** any order that the computer carries out immediately. In LOGO, if you tell the small "turtle" on the screen to move FORWARD 20 places, it will do that right away. In BASIC, if you give the computer the command PRINT 3*4, your computer will immediately answer *12*. But if you use a line number:

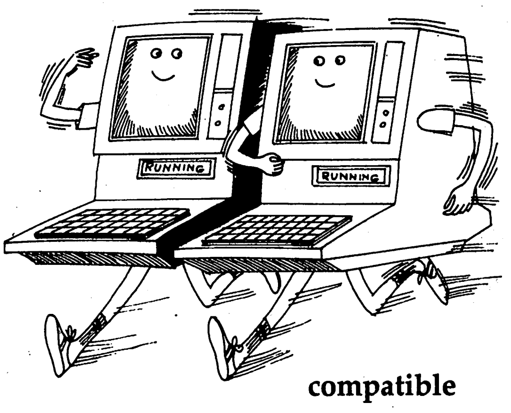
```
10 PRINT 3*4
```

the computer will not give the answer until you type the command RUN. **2.** anything you tell the computer to do, now or later. When you type a program into the computer, you often include instructions to READ, GOTO, and PRINT. Sometimes these instruction words are called commands, even though the computer waits until you finish typing the whole program and then does everything at once.

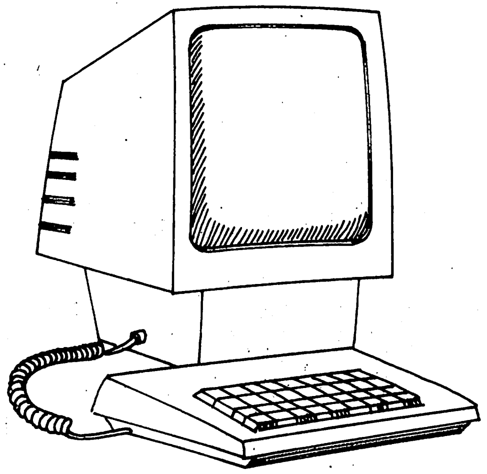


command

**compatible** (kəm pat' ə bəl): able to work together with something else. A computer program, for example, is "compatible" with a computer that can run it. Two computers which can run the same programs are said to be compatible with each other. There isn't a whole lot of compatibility among different computers. You find big differences in their operating systems. That's why a PET computer, for example, doesn't know how to "operate" (run) a game for a TRS-80 computer. Before you buy a disk or tape, you have to ask: (a) Was this program written for my brand and model of computer? (b) Is it compatible with my computer's operating system? The same idea applies to printers and other attachments. See **operating system**.

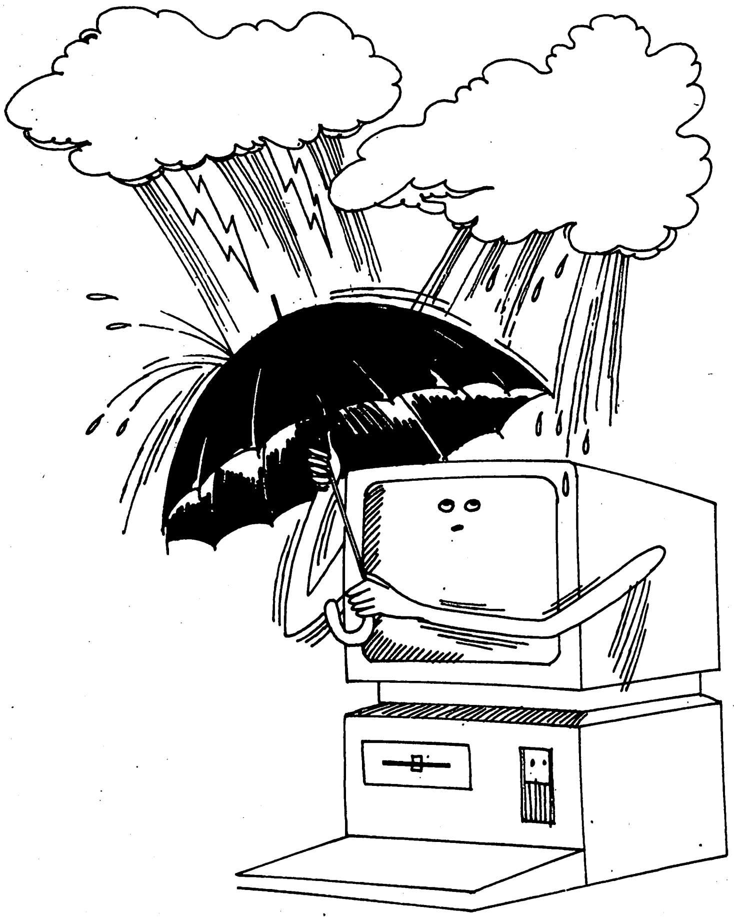


compatible

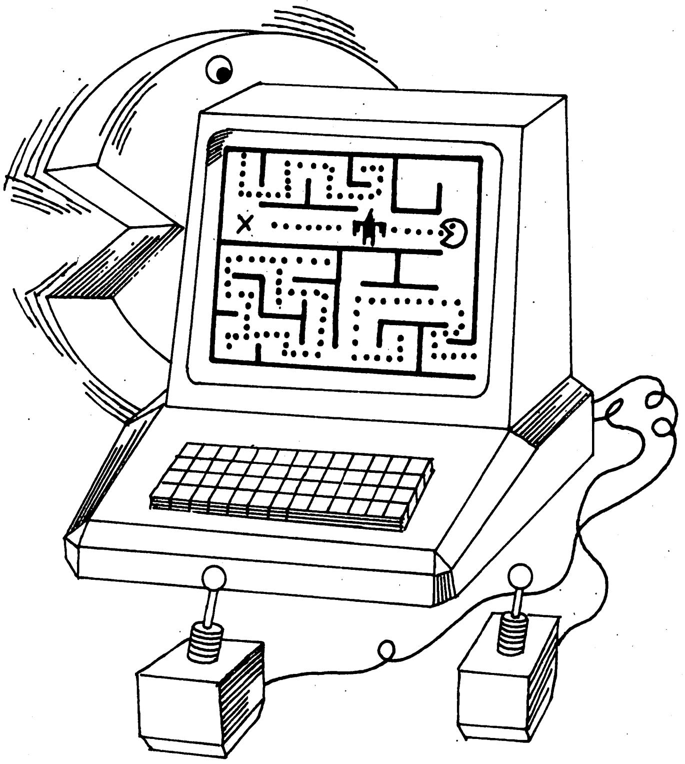


computer

**computer** (kəm pyū′ tər): what this dictionary is all about! A computer can be defined in the following four ways: **1.** *by the work it does.* Every computer has three main functions: (a) It can take information from outside itself ("input"); (b) it can work on that information by following a set of instructions; (c) it can show, or display, results ("output"). **2.** *by the kind of information it handles.* Computers can be classified as analog or digital. (a) An **analog computer** takes continuous input about conditions such as heat or light. It combines these inputs and produces a coded electrical signal. The signal changes as soon as those conditions change. (Analog computers are used for such things as adjusting air pressure in airplanes or for steering ships.) (b) A **digital computer** "chews" numbers and uses them to solve very tough problems (including how to make Pac-man chew). It has a memory it never loses, and a memory you can erase. Almost all the entries in this dictionary deal with digital computers. **3.** *by its size (power) and by its price.* A digital computer is either a "mainframe," a "mini," or a "micro." (a) Forty years

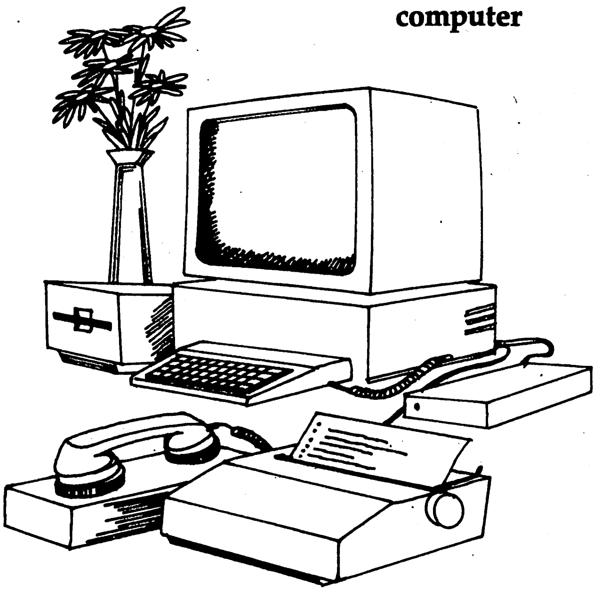


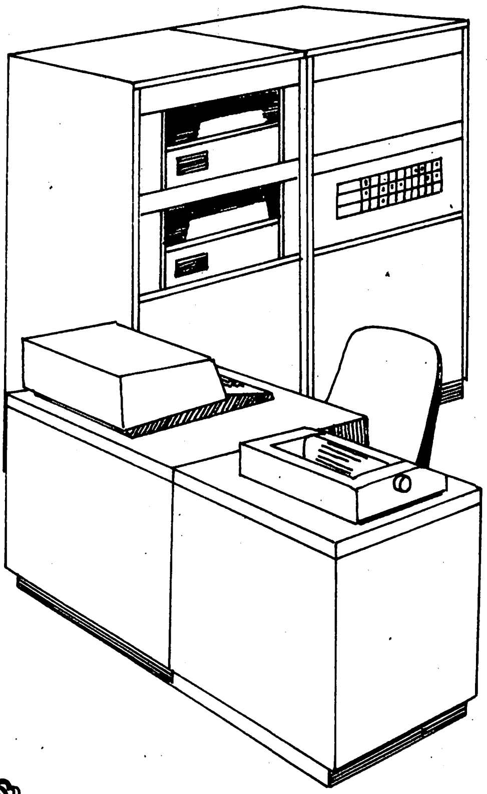
analog computer

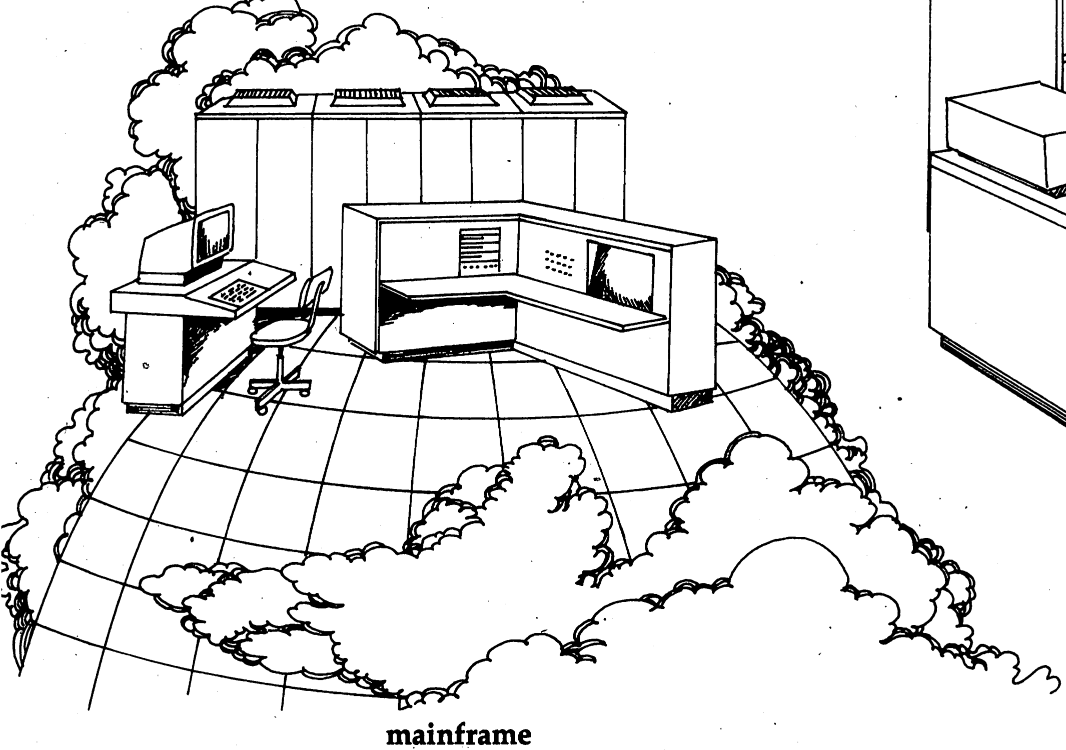


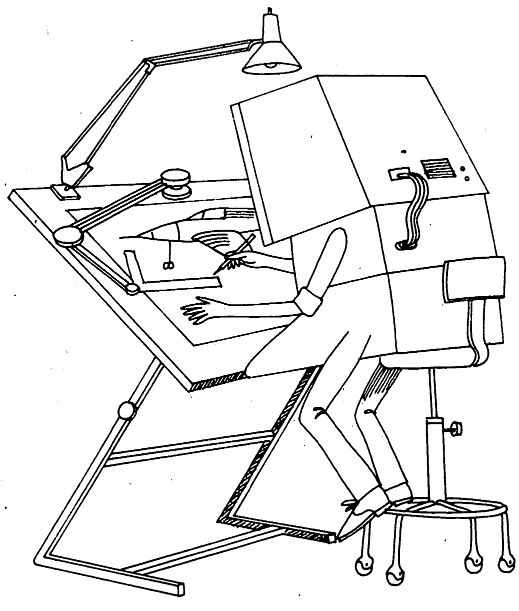
digital computer

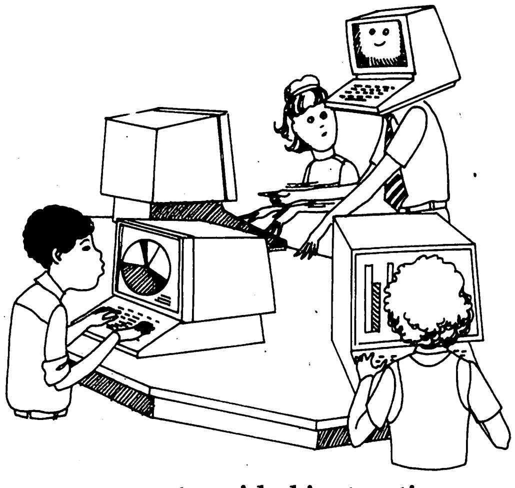
ago, large **mainframes** were the only size that a computer could be. They are still the largest size, and can handle more than 100,000,000 instructions per second. PER SECOND! Mainframes cost at least several hundred thousand dollars. They are used for big jobs by large organizations like national weather bureaus and international banks. (Mainframes are also called general-purpose computers.) (b) A **minicomputer** is the next largest in size and price. It does fairly large jobs, like making out payrolls. Minis often work on "time sharing." This means that people at several keyboard terminals can use the same central computer at the same time. Minicomputers cost $20,000 or more. (c) Finally, there is the **microcomputer** (also called a personal, or "desk-top" computer). It's the smallest and the cheapest kind. **4.** *by its brand name.* Computer brands range from huge IBM mainframes to the small Timex-Sinclair personal computer (with Apples, Pets, and other nice computers in between). See . . . all the terms in this book!

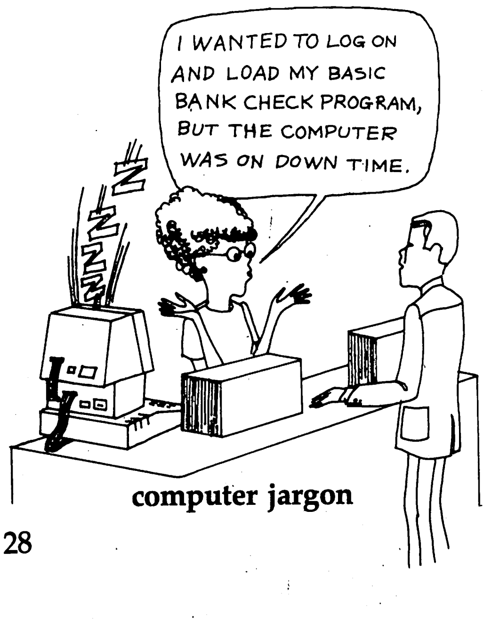

**microcomputer**


**minicomputer**


**mainframe**


computer-aided design


computer-aided instruction


I WANTED TO LOG ON AND LOAD MY BASIC BANK CHECK PROGRAM, BUT THE COMPUTER WAS ON DOWN TIME.
ZZZ
computer jargon

**computer-aided design** (kəm pyū′ tər ād′ əd di zīn′), **CAD**: the use of computers to help engineers design cars, spaceships, hospital equipment, and other computers. For example, a computer can tell if an airplane-wing design would support a plane with 300 passengers. It's better to find that out on a computer than on a plane. See **computer-aided manufacturing.**

**computer-aided instruction** (kəm pyū′ tər ād′ əd in struk′ shən), **CAI**: the use of computers to help students learn, practice, and increase skills. When you use a CAI program, you have to solve problems or answer questions that the computer gives you. The computer "checks" your answers and lets you know if something is wrong. An advantage of CAI is that it allows a student to learn at his or her own pace. (CAI also stands for "computer-assisted instruction".) See **computer-aided learning.**

**computer-aided learning** (kəm pyū′ tər ād′ əd lur′ ning), **CAL**: the use of computers to explore new ways of thinking about problems and how to solve them. The LOGO language is used for CAL. Beginners in LOGO learn to write programs just by experimenting. And they learn science and math as well. See **computer-aided instruction, LOGO, turtle, program.**

**computer-aided manufacturing** (kəm pyū′ tər ād′ əd man′ yə fak′ chər ing), **CAM**: the use of a computer to help manufacture certain products. Computers can measure the time it takes to produce one item. They can test the weight and color and shape of a button. And they can tell a manufacturer how many new umbrellas to make in a season. See **computer-aided design.**

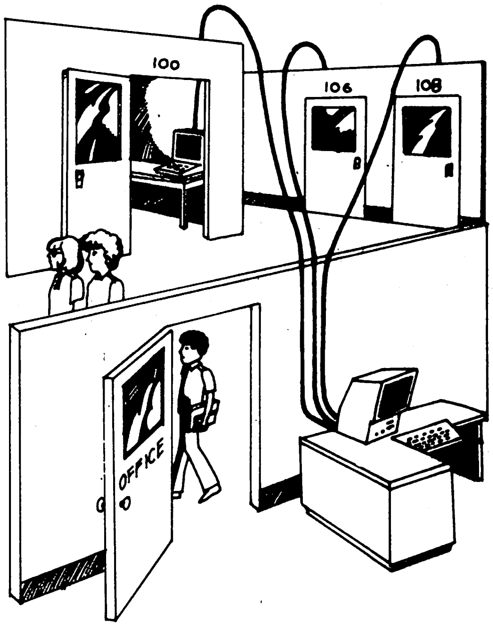
**computer code** (kəm pyū′ tər kōd): see **language** (definition a).

**computer jargon** (kəm pyū′ tər jär′ gən): slogans, abbreviations, and other computer-related terms that make computers seem like a shipment from Mars. Take this example: "I wanted to log on and load my BASIC bank-check file, but the computer was on down time." What's the subject? Trees? Guns? Music? No. This computer jargon simply means "I wanted to find my bank balance, but the computer wasn't working." See **metaphor, acronym, mnemonic.**

**computer language** (kəm pyū′ tər lang′ gwij): see **language.**

**computer literacy** (kəm pyū′ tər lit′ ər ə sē): an understanding of (a) what computers are for, and (b) how they affect our lives. Literacy means the ability to read and write information. Computer literacy means the ability to use a new source of information—the computer. You don't have to know how to write programs to be "computer literate." But computer specialists say that you should be able to think about these big questions: How will computers affect our jobs? How will they affect our privacy? What can we do with computers to improve life on our planet?

**computer-managed instruction** (kəm pyū′ tər man′ ijd in struk′ shən), **CMI**: a way of using the computer in a large school or school system. CMI involves hooking keyboard terminals in different classrooms to one central computer. This central computer (a) feeds lessons in math, English, career planning, and so on, to students at different terminals; (b) keeps a daily record of students' work; and (c) keeps all the other records a school might want it to. The purpose of CMI is to help students learn skills at their own speed. When CMI frees teachers from record-keeping, they have more time for the kind of lessons a computer cannot give. See **computer-aided learning.**



computer-managed instruction

**computer output** (kəm pyū′ tər out′ put′): see **output.**

**computer program** (kəm pyū′ tər prō′ gram): see **program.**

**computer science** (kəm pyū′ tər sī′ əns): the study of how to design, build, and use computers.

**computer word** (kəm pyū′ tər wûrd): see **word.**

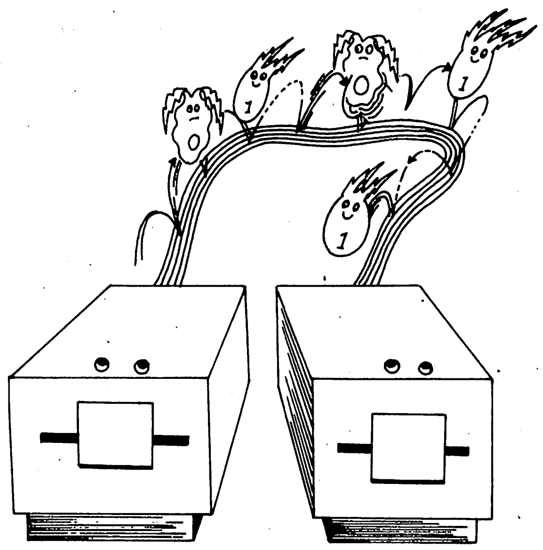
**conditional branch** (kən dish′ ən əl branch): see **branch** (definition 2).

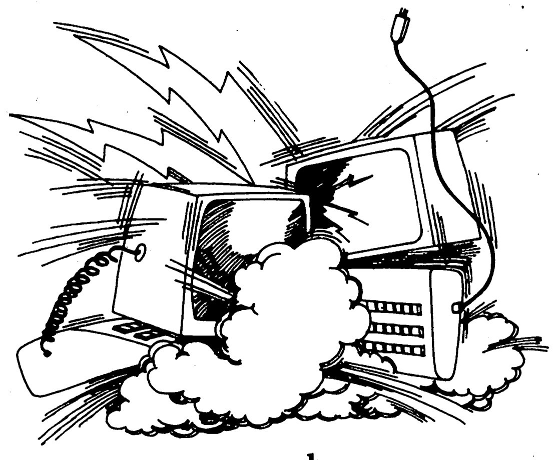
**conditional jump** (kən dish′ ən əl jump): see **branch** (definition 2).

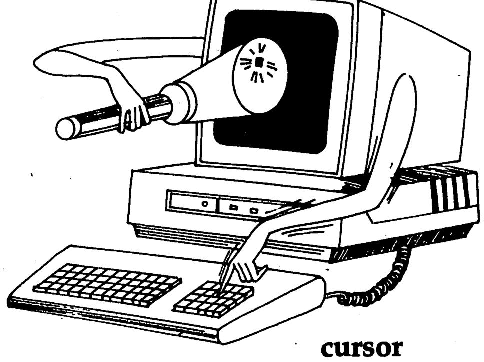
**conditional transfer** (kən dish′ ən əl trans′ fər): see **branch** (definition 2).

**console** (kon′ sōl): the part of a computer that you use to "talk" to it and get replies. In a large computer system, the console can include a panel with switches and lights, a screen, a printer, and a typewriter. In a small computer, your keyboard is your "console," and you can usually see replies on a screen.

console


copy


crash


cursor

**control key** (kən trōl′ kē′), **CTRL**: on most keyboards, a key that allows you to interrupt something the computer is doing. When you use CTRL, you may have to touch one other key, too. For example, you might press both the C key and CTRL to stop a BASIC program while it is running. The user's manual for each computer tells you how to use the control key.

**Control Program for Microcomputers** (kən trōl′ prō′ gram fər mī′ krō kəm pyū′ tərz), **CP/M**: see **operating system.**

**control unit** (kən trōl′ yū′ nit): part of a computer's central processing unit. The control unit reads your instructions and assigns work (such as doing arithmetic, sorting answers, taking data from a tape or disk) to other areas of the computer. See **central processing unit.**

**copy** (kop′ ē): **1.** in programming, to transfer information. When you tell the computer to READ data, it takes information from one place (such as a tape) and "copies" it into its active memory. **2.** to duplicate a tape. It's a good idea to make backup copies of your tapes and disks.

**COPY**: in word processing, a key you press to repeat words in another part of the same text. If you quote a poem at the beginning of a composition, and you want to repeat it at the end, you can COPY it. That is, you can tell the computer to read it in one place and reprint it in another. See **MOVE, edit.**

**CP/M**: see **operating system.**

**CPU**: see **central processing unit.**

**CR**, Carriage Return: see **RETURN.**

**crash** (krash): a major breakdown of a computer system. A crash can happen either to the computer itself or to programs it is working on.

**CRT**: see **cathode-ray tube.**

**CTRL**: see **control key.**

**cursor** (kůr′ sər): a small light on your computer screen that shows you where your next keystroke will appear. Most computers have a key or button that you press to move the cursor around. This allows you to go back over your work and change it. See **arrow.**

## D

**daisy-wheel printer** (dā′ zē hwēl′ prin′ tər): see **printer**.

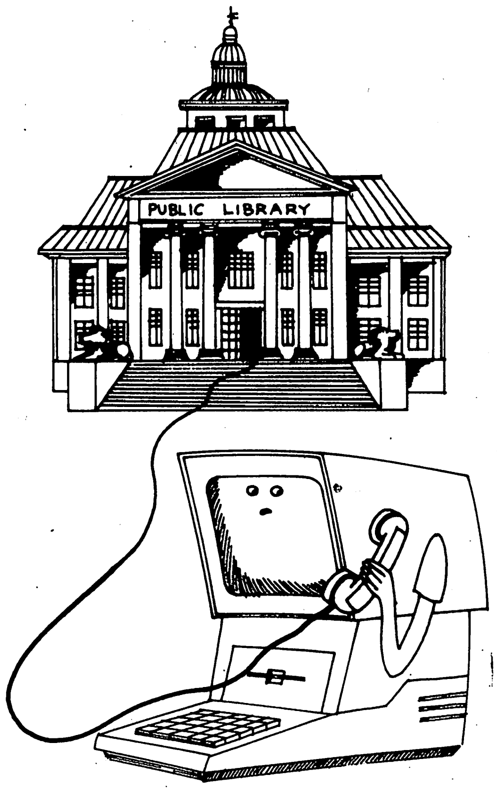
**data** (dā′ tə): **1.** in general, any information you give your computer, or that a computer works with. This includes instructions for a game, lists of names, facts you get from another computer. **2.** within a program, numbers your computer works with. These can be numbers from a tape, numbers you type, or answers that the computer gets when it works on a problem. In the program below, the computer gets data in two ways. (A = allowance; S and S1 = money spent; T = total; R = what's left)

```
10 REM ALLOWANCE PROGRAM
20 LET A = 5.00
30 LET S = 1.75
40 LET S1 = 2.00
50 LET T = S + S1
60 LET R = A − T
70 END
```

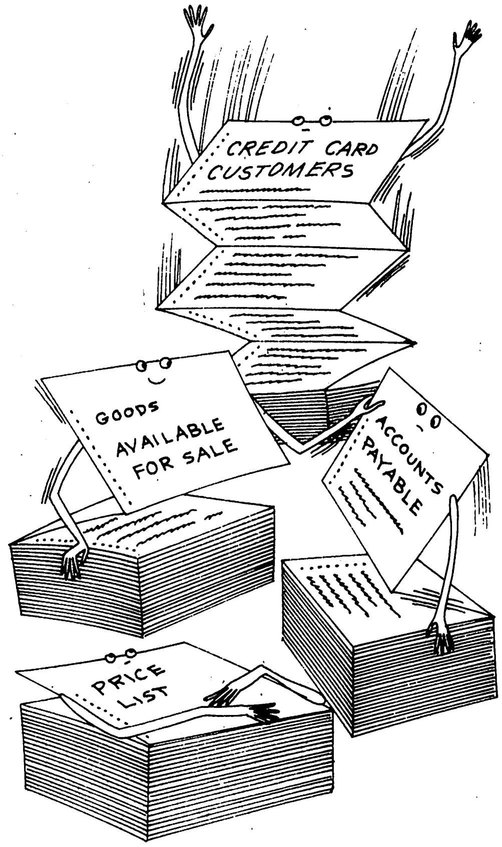
In lines 20 – 40, you give the computer some data. In line 50, you tell the computer to work on it. In line 60, you tell the computer to use a piece of data (T) it found by itself.

**DATA**: see **READ . . . DATA**.

**data bank** (dā′ tə bangk): a huge source of information that a computer can tap into, with the right password and special equipment. If your public library were a data bank, for example, here's how it might work: You could (a) hook up your computer to your telephone; (b) call the public library; (c) type your special password; and (d) read a new book, page by page, on your computer screen. Public libraries aren't data banks, yet. But for a fee, you can get other kinds of information from computer services around the country. One data bank, The Source, gives up-to-date news, travel ideas, and education services. You can even "chat" with other computer owners who use The Source. The Dow Jones News/Retrieval Service gives stock-market and business news. To use a data bank on a personal computer, you need a "modem." That's an attachment that changes computer signals to telephone signals (and vice-versa). Data banks are sometimes called data bases, but there is a difference. See **data base, videotext, teletext**.



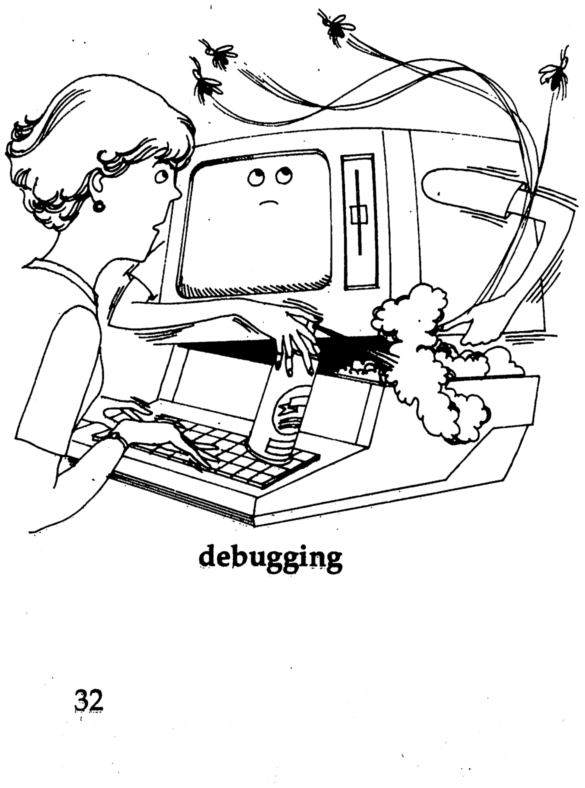
**data bank**



data base

**data base** (dā′ tə bās): all the facts that a company has to keep track of to do its business. This might include (a) names and addresses of customers with credit cards; (b) totals that each customer spends on products; (c) lists of all the goods the company sells; and more. From this data, the company could get its computer to make lists of customers who buy only at holidays or who are late in paying bills. Using the same data base, the sales manager can mail ads about holiday sales to one group of customers. And the account manager can mail reminders about late payments to others. Some computer specialists say there is a lot of private information about us on business and government data bases. Should the use of data-base information be controlled? ("Data base" is sometimes used to mean "data bank," but there is a difference.) See **data bank.**

**data processing** (dā′ tə pros′ es′ ing), **DP**: by a computer, the arrangement of data in a way that makes them more useful to you. If your group ran a telethon to raise money for a new ambulance, you would need to keep donors' names in order and a fast, accurate total of the money they promised to give. A computer could "process" this information by putting each name into alphabetical order, and doing instant addition—whether the pledges came in tens, hundreds, or thousands of dollars. The computer doesn't "know" what these data mean. But it can process thousands of numbers into a total that you'll understand . . . perfectly.



debugging

**debugging** (dē′ bug′ ing): getting rid of mistakes in the design of a computer; correcting errors in the way a program is written. Computer specialists can spend months getting rid of the bugs in a new computer model. It can take weeks to debug a set of program instructions, especially if there are many steps. See **bug, test.**

**decimal system** (des′ ə məl sis′ təm): a counting system that uses 10 numbers — 0 to 9. Compare this to the binary system. See **binary system.**

**decision** (di sizh′ ən): your computer's selection of the next step it will take when it has at least two choices. Suppose you are training for an athletic event. You want the computer to keep track of your exercise time each day and to report on your progress once a month. You're hoping to improve on your best time so far: 3600 minutes a month.

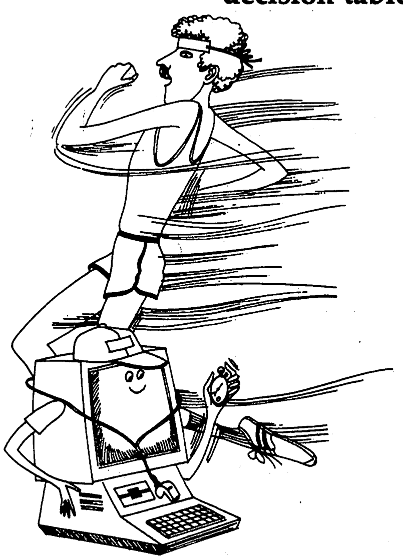
Here's a part of your BASIC program:

```
    460 IF T < 3600 THEN 490
    470 PRINT "GREAT WORK!"
    480 GOTO 500
    490 PRINT "YOU'RE SLIPPING!"
```

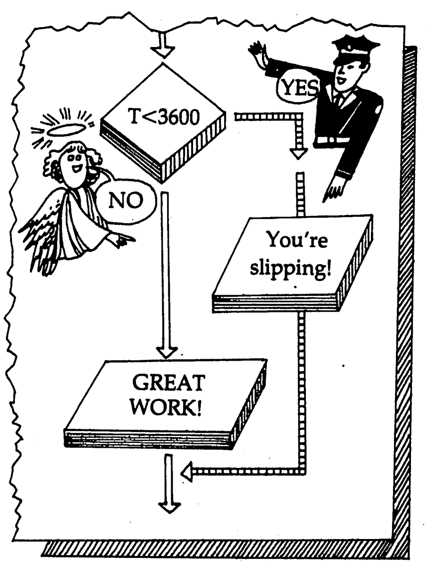
On line 460 you tell the computer to look into its memory for your total exercise time during the past month. If your total (T) is under 3600, your computer will jump to line 490, and print a warning. But if you did break your time, the computer just goes on to line 470 and gives you a pat on the back. The computer's "decision" involves (a) searching its memory, (b) comparing numbers, and (c) reacting to what it finds. You make the decision. See **branch, decision table, flowchart.**

**decision box** (di sizh′ ən boks): a diamond-shaped box that is a part of some "flowcharts." (A flowchart diagrams instructions for a computer program.) The decision box shows where you want the computer to carry out a "test." It must indicate (a) the "yes-no" test you want the computer to make; and (b) the next step the computer should take if the test comes out "yes" or if it comes out "no." These yes-no results point in different directions from the decision box. See **flowchart, branch.**

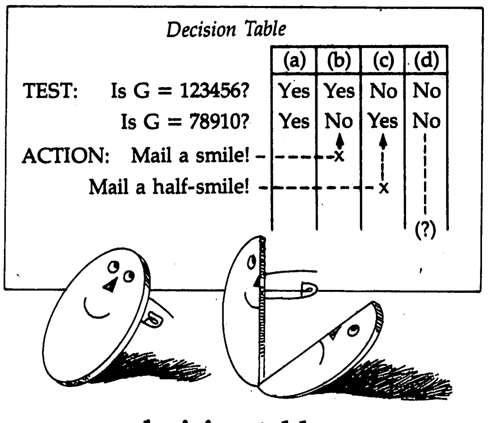
**decision table** (di sizh′ ən tā′ bəl): a chart you make to help you plan a computer program. You use it to make sure you write all the instructions you need. Imagine a silly contest: You put two numbers into a computer. Anyone who guesses 123456 wins a smile button. Anyone who guesses 78910 wins half a smile. The computer will keep track of guesses (G) and mail results to everyone. To plan the program for this silly contest, you draw a decision table. In it, you list the two winning numbers. Next to them you put columns of "yes-no" answers—all the answers a computer could get when it checks each guess. ("Is this number 123456?" "No." Is it 78910?" "Yes.") You check to see if these yes-no columns make sense. In the table on this page, column (a) doesn't make sense. (If someone's guess is 123456, then it can't also be 78910.) Cross out column (a). Columns (b) and (c) are OK. The "x's" tell what your computer mails to each winner. But what about column (d)? People who don't guess a winning number want to know results, too. So your computer will need another instruction. (Maybe mail a frown?) See **decision, branch, flowchart.**



**decision**



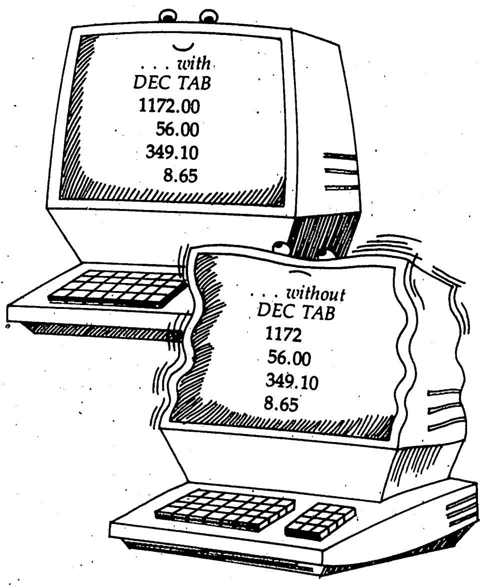
**decision box**



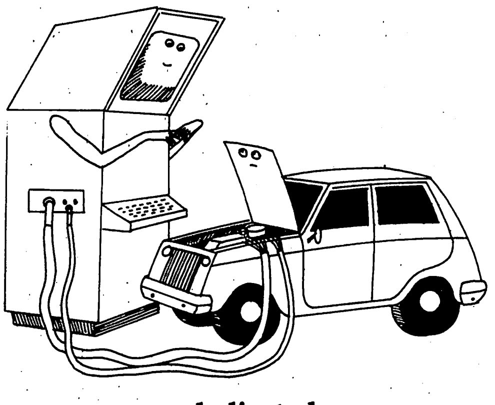
*Decision Table*

| | | (a) | (b) | (c) | (d) |
|---|---|---|---|---|---|
| TEST: | Is G = 123456? | Yes | Yes | No | No |
| | Is G = 78910? | Yes | No | Yes | No |
| ACTION: | Mail a smile! | | x | | |
| | Mail a half-smile! | | | x | |
| | | | | | (?) |

**decision table**



DEC TAB



dedicated

**DEC TAB** (dek tab), **DEC**imal **TAB**: in word processing, a key you touch to make the computer line up numbers by decimal points. The illustration shows what happens when you use DEC TAB—and when you don't!

**dedicated** (ded′ ə kāt′ əd): **1.** set aside for one purpose or one user. Suppose your school has a large computer system, with several keyboard terminals. Your principal's terminal might be dedicated, set aside for record-keeping. **2.** made for one purpose only. A "dedicated" computer is a computer designed to do one specific job, like testing your car or processing words. A doctor might order a dedicated computer just to handle her research. This research computer would probably be a whiz at science problems but it might not be able to run even a simple game. (Like humans, computers can't be good at everything . . .)

**DEF FN-, DEF**ine **F**unctio**N**: in BASIC, the letters you use to put a new rule into your computer's memory. Suppose you're in charge of reporting the average daily hours that volunteers work each week at a day-care center. You invent a rule for finding this average, and then you put the rule into your computer's memory this way:

```
DEF FNA(X) = (M + T + W + H + F) /5
```

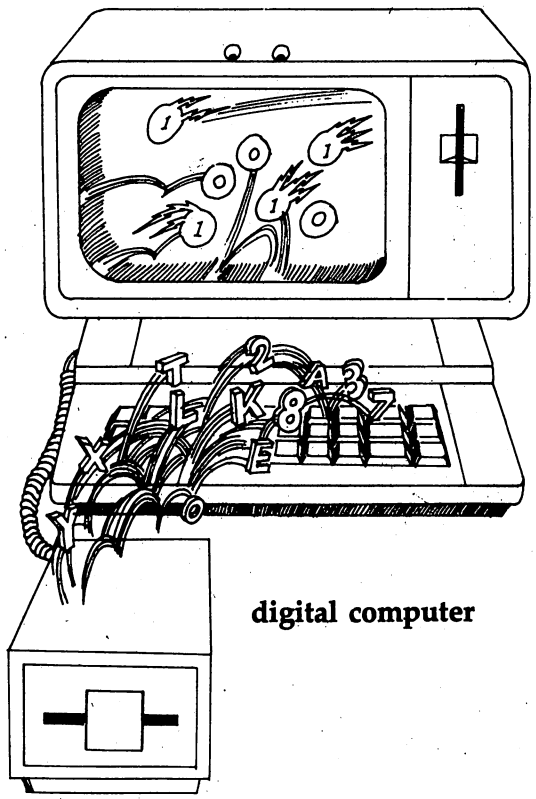
DEF FN means "take this into your memory." The A stands for this rule (a rule for averages). The (X) takes the place of any particular week that you use this rule (the 10th week, or the 15th, etc.). The rule itself says to the computer: Add all five days' totals; then divide that sum by 5. See **user-defined function.**

**DEL, DEL**ete: the code that tells the computer to "forget" the previous number, letter, or symbol. In BASIC, the command DEL 30 tells the computer to remove line 30 from a program. See **line number** for another way of doing this.

**DELETE** (di lēt′): **1.** in word processing, a key that tells the computer to remove a word, line, or paragraph from what you're typing. See **space bar, INSERT, edit. 2.** in BASIC, your command to the computer to get rid of a program (a set of instructions) in its memory. Suppose you have two forms of a Spanish-language program in your computer. You decide that SPANB is better. You want to get rid of SPANA to use that space for a new program. You tell the computer: DELETE SPANA. Think twice before using DELETE. Once you type it, Z-Z-Z-A-P! No more program!

**dialect** (dī′ ə lekt′): a slightly different version of a standard computer language. Dialects are invented to go with different brands of computers. PET BASIC, for example, is used with Commodore computers. APPLE BASIC is used with another major computer brand. Dialect commands are sometimes different from commands in the standard language. You need to check the user's manual of your computer for these differences.

**digital computer** (dij′ ə təl kəm pyū′ tər): a computer that works with specific numbers (including words that are coded by numbers). A digital computer can get its numbers from punched cards (it "reads" the holes punched in these cards), a magnetic tape or disk, a telephone, a television signal, or you — when you type numbers at the keyboard. A digital computer can do two tasks with numbers: It can calculate with or combine them, as in "2 + 5." It can also compare them as, for example, when it checks its memory to see if "A = 9." Your brain works like a digital computer when you count change at the store or look up a word in a dictionary. See **analog computer, computer.**



digital computer

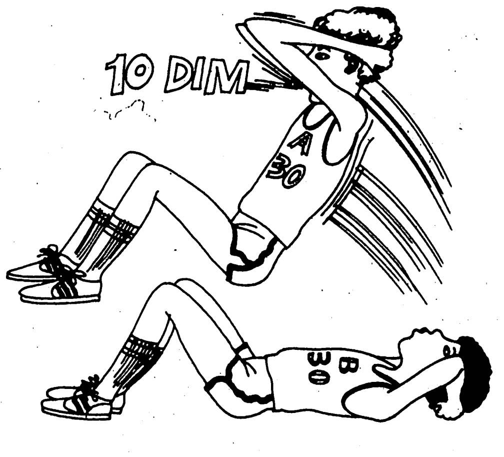
**DIM statement** (dim stāt′ mənt), **DIMension** statement: in BASIC, your way of getting the computer to save space in its memory for a list of numbers. Suppose you want to keep track of the number of situps you and your pal do every day this month. You would include the following DIM instruction in your program:

```
10 DIM A(30), B(30)
```

The program line number is 10. "DIM" is the code word for "dimension" (space). You are "A." Your pal is "B." And you now have 30 spaces each in the computer. Here is how you would use a DIM statement to have the computer save spaces for 30 names:

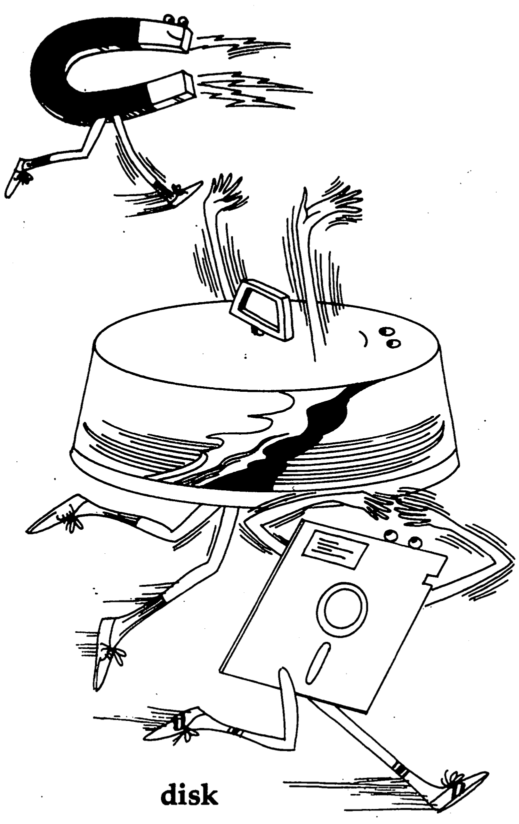
```
10 DIM N$(30)
```

The dollar sign ($) tells your computer that the 30 spaces will be used for "strings" of letters, numbers, etc., not just numbers. See **list, string** (definition 2).

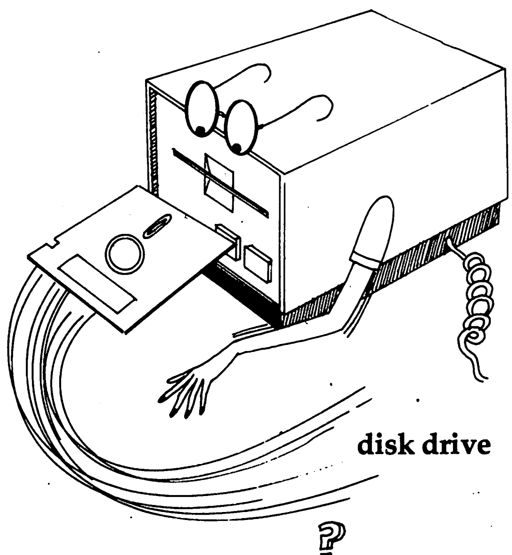


DIM

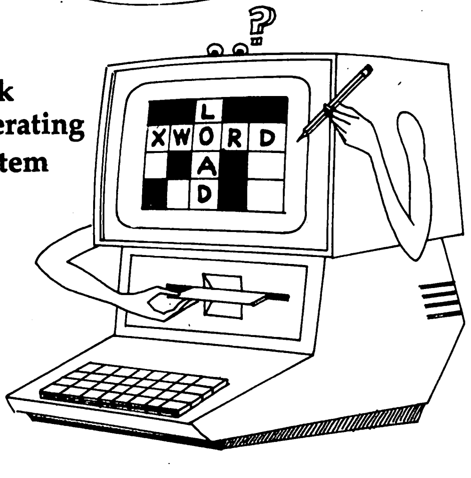
**direct access** (di rekt′ ak′ ses): the ability to find any fact in the computer's memory without delay. When a computer takes a new fact into its memory, it gives that fact an "address." This address is a number code for where the fact is. It works like the zip code in your home address. This address system gives your computer direct access to any piece of data in its main storage area or on a magnetic disk. See **disk, address.**



disk



disk drive



disk operating system

**disk** (disk): a flat, circular object on which you can store computer program instructions and data. It is the best kind of back-up memory a computer can have. The disk's surface is coated with a rust-like material that makes it magnetic. A disk can be easily damaged if dust or a fingerprint gets on it, or easily erased if it is near a magnet. For this reason, the disk always wears a protective jacket, even when it's in use! Disks come in several sizes. For large business computers, there are *hard disks* of aluminum, up to 14 inches in diameter. For small microcomputers, there are soft plastic disks, usually 5¼ inches in diameter. Soft disks have many other names: *diskettes, flexible disks, floppies,* and *minifloppies!* All disks have invisible tracks, like circles in a bull's-eye target (only much closer to one another.). The whole surface of the disk is divided into "sectors" (like pieces of pie). If you need a program that's stored in the middle of a disk, your computer can find it in a split, split second, by track and sector. You can buy disk programs "ready-made" or blank disks to record your own programs. You need a disk drive to run disks for your computer. See **disk drive, storage.**

**disk drive** (disk drīv): a device that puts computer data onto a magnetic disk, and later reads it back into the computer's memory. The disk drive is like a secretary for the computer, taking information, reading it back. Instead of a notepad, the drive "writes" on a disk (a flat, round piece of aluminum or plastic) which has been inserted into it. The drive has a "read/write head" that creates tiny magnetic "dots" on the spinning disk. When the disk drive "reads" a disk, it just reverses the process—reading these dots from the disk so the computer can change them into electric pulses ("bits"). See **disk, storage.**

**diskette** (dis ket'): see **disk.**

**disk operating system** (disk op' ə rā' ting sis' təm), **DOS:** a set of program instructions that tell a computer how to use a magnetic disk drive. Some computers have their own built-in DOS. For other brands, you have to purchase one if you want to store data on disks. A widely used DOS is the CP/M (Control Program for Microcomputers). The DOS makes life simple for a disk-user. Suppose you want to use a crossword puzzle program that you put onto your disk a long time ago. All you have to do is type in a simple command, such as LOAD XWORD. Your DOS will search its own record of where that program is. Then it will get the disk drive to transfer your crossword program to the computer's memory. Seconds later, your puzzle is on the screen. See **operating system, disk, disk drive.**

**display** (dis plā'): in computing, a "dark-light" image created by a weak electric current and used to show data. Usually, computer display appears on a TV-like screen. When you use a small pocket calculator, you see a display through a "sandwich" of glass. See **cathode-ray tube, visual display unit.**

**documentation** (dok' yə men tā' shən): **1.** remarks you put at the beginning of a program to remind yourself later of what it's about. Documentation for a program in BASIC might begin this way:

```
10 REM PROGRAM FOR SPELLING GAME
```

REM (for "remark") simply tells your computer to print the comment each time it lists your program's instructions. See **REM, program.** **2.** information a computer manufacturer gives to buyers, usually in a user's manual. It should include a general description of your computer and its parts, instructions for installing it, directions for using it, and advice about problems you might run into as a first-time user.

**DOS**: see **disk operating system.**

**dot matrix printer** (dot mā' tricks prin' tər): see **printer.**

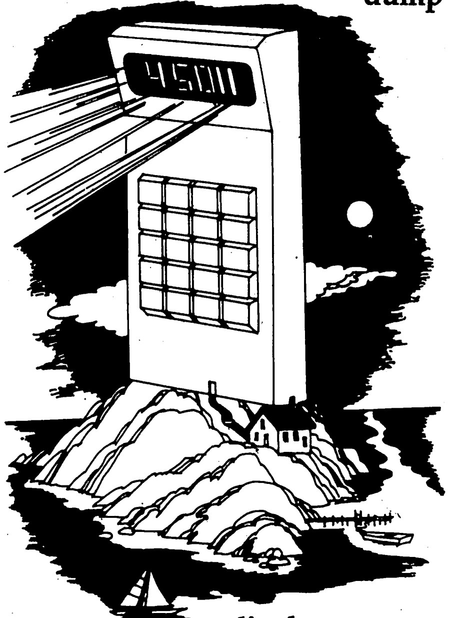
**down time** (doun' tīm'): the time when a computer system is not at work or is being repaired. See **up time.**

**DP**: see **data processing.**

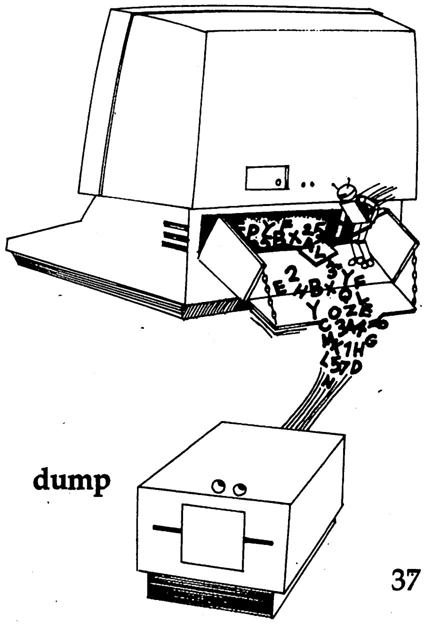
**DRAW** (drô): in LOGO, a command that gets the computer ready to show drawings on its screen. You "draw" by typing directions for a small screen figure, called a "turtle." See **NODRAW.**

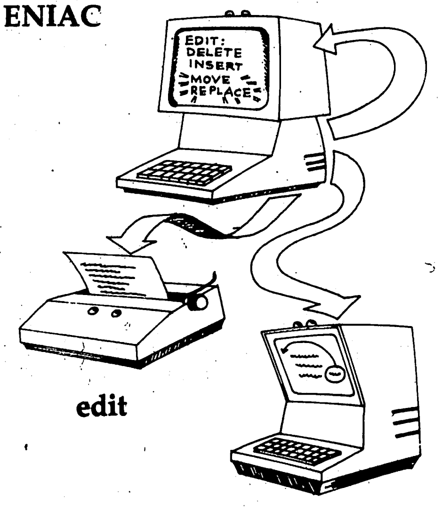
**dumb computer** (dum kəm pyū' tər): see **unintelligent computer.**

**dump** (dump): **1.** to transfer data from the computer's active memory to a disk or tape. Business, bank, and government records are "dumped" regularly by their computers. Then, if the computer breaks down, only the last part of a job is lost. **2.** to print a list of program steps, after the computer carries them out. Program writers often have to dump their work, to check for errors. (This is also called "debugging.") See **debugging.** **3.** to transfer data from magnetic tape to magnetic disk, or vice versa. See **storage, disk, tape.**

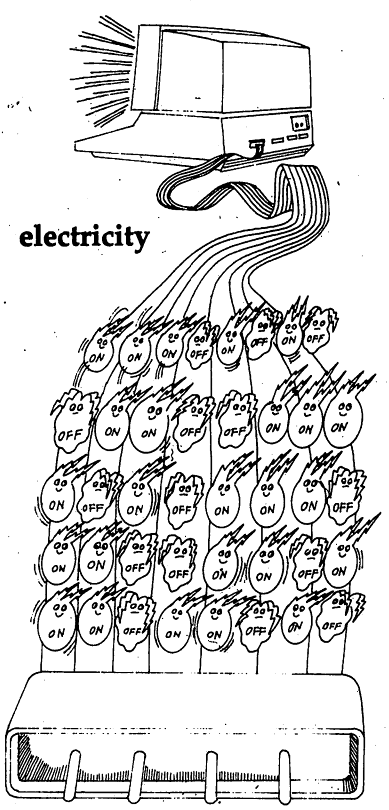

display

documentation

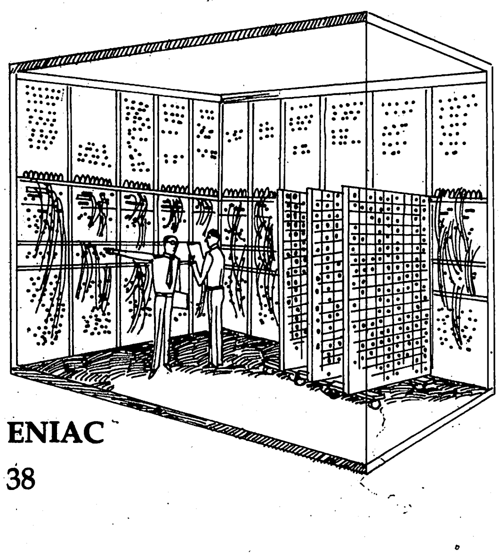

dump



edit



electricity



ENIAC

## E

**edit** (ed′ it): **1.** by a computer, to get data ready to "move." Ready to move where? To your computer screen, to a printer, or to another computer. These moves usually involve adding an extra code to computer data. In editing, your computer is like a postal worker, making sure every piece of information gets shipped in the right direction. **2.** by you, to change what you have already typed and can see on the computer screen. To edit on a computer, you can add new words (insert), or take them out (delete). You can move lines from one place to another, or keep a copy of them in two places . . . or three, or four. See **COPY, DELETE, INSERT, MOVE**.

**electricity** (i lek′ tris′ ə tē): where a computer gets its energy! Actually, the electric current in a small computer is weak. Its flow is controlled by semiconductors. A computer works by keeping tiny electric pulses in rows, all moving at the same speed. Humans can then invent codes for the "on-off" pulses ("bits") marching around. Well, they don't exactly march. They fly at the rate of millions per second . . . but in rows, always in rows. See **semiconductor, circuit, transistor, ASCII.**

**END** (end): in BASIC, the last instruction you give your computer when you type a program. Some computers do not need an END command.

**end-of-text** (end uv tekst), **ETX**: a code meaning that the letter ("character") just before "ETX" was the last one in the message.

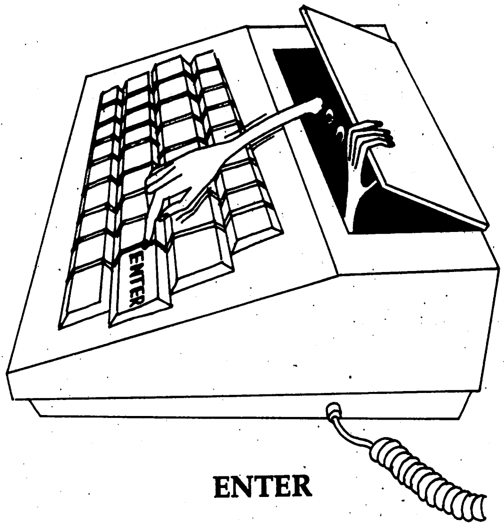
**end-of-transmission** (end′ uv trans mish′ ən), **EOT**: a code meaning that the computer has just finished sending a message to a distant point.

**ENIAC** (en′ ē ak′), **Electronic Numerical Integrator And Calculator**: the first successful modern computer in the United States. ENIAC was built at the University of Pennsylvania in 1946. It was the size of a room and weighed 30 tons! ENIAC used 18,000 vacuum tubes to carry data in electric pulses. These tubes produced a great deal of heat and broke down at the rate of one every seven minutes. But when it worked, ENIAC could add 5,000 numbers a second. The computer age was here!

**ENQ, ENQuiry:** a computer code for "Who are you?" When a computer is about to send a message to a keyboard terminal someplace else, it opens with ENQ. That is, it asks for an I.D. from the other side. Spies and double agents are cut off immediately.

**enter** (en' tər): **1.** to start running a program at some point after its first step. (It's a little like singing just the middle part of a song.) **2.** to send a message from a keyboard terminal to the main computer. **3.** to use the ENTER key. See **ENTER.**

**ENTER:** on some computer keyboards, the key you touch to show you are satisfied with the line you have just typed. Your computer will not do anything with a line you type until you tap the ENTER key. This gives you time to check each line before locking the statement in. On many computer keyboards, the key that does this is called RETURN. Same thing. See **RETURN.**



ENTER

**EOT:** see **end-of-transmission.**

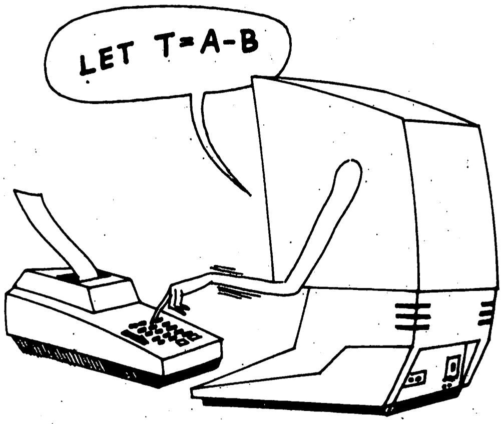
**error** (er' ər): see **bug, SYNTAX ERROR.**

**ETX:** see **end-of-text.**

**execute** (ek' sə kyūt'): to carry out an instruction or run a program. *It took me six months to write that program. My computer executed it in two seconds!*

**EXECUTE:** in word processing, a key you touch to make the computer carry out a command. For example, if you want to move a sentence from one place to another, you touch MOVE, then EXECUTE. It's as though you tell the computer, "I want you to do this. Yes, I do."

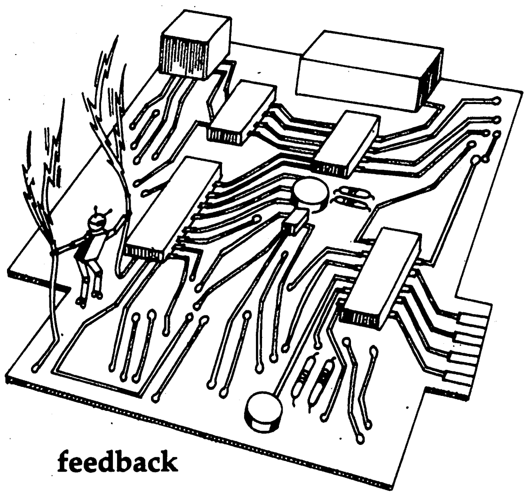
**exponentiation** (ek' spə nen' chē ā' shən): the process of multiplying a number by itself one or more times. Suppose you want to multiply 10 by itself 5 times. In math, a short way of stating this is $10^5$—or 10 to the fifth power. In BASIC, you would write this instruction a different way: 10↑5. Either way, the answer is 100,000. See **arithmetic operator.**

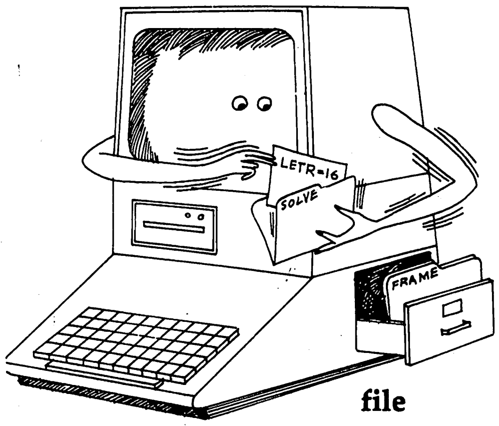


expression

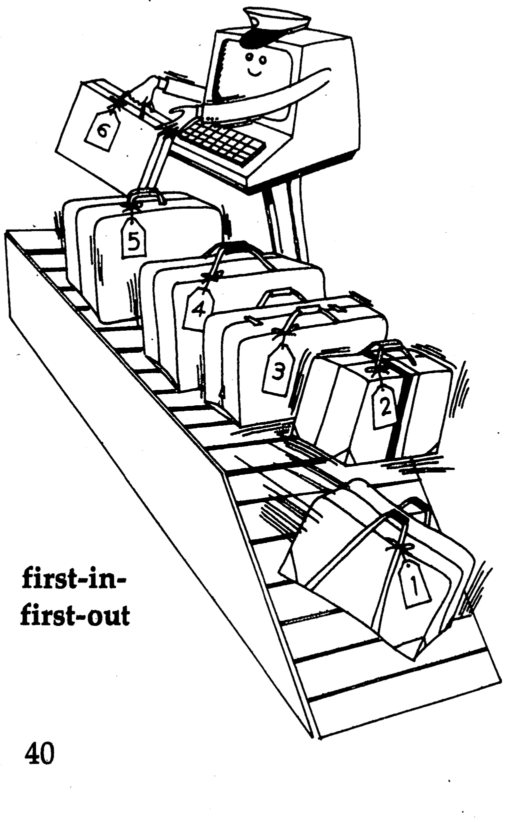
**expression** (ik spresh' ən): numbers and/or letters, joined by arithmetic symbols that the computer can read and carry out. In the statement LET T = A − B, the expression is A − B. (The computer would have to search its memory for the values of A and B before subtracting.)

F


feedback



file



first-in-first-out

**feed** (fēd): **1.** to move paper through a computer printer. **2.** to move hole-punched cards through a "reader" to put data into a computer. See **Hollerith code. 3.** to give your computer a program to work on. There's nothing like a little snack of numbers and letters to make your computer happy. See **number crunching.**

**feedback** (fēd' bak'): a computer's report to itself on how it is working. A computer "reads" and "talks" in code—in pulses of electricity. If the electric flow at any point is too slow or fast, the code breaks down. Then nothing makes sense to the computer or you. A good computer has feedback points where it checks up on the electric flow. It sends a signal back along the line to slow down or speed up. It's like what your body does for you when you exercise. If you get too heated, you sweat. If your muscles need oxygen, they cramp. Feedback.

**FIFO** (fī' fō): see **first-in-first-out.**

**file** (fīl): a set of coded records that your computer tucks away under a single label. You invent the labels. FRAME might be the file name of all your friends' addresses. SOLVE might hold programs you've written. Your computer knows how to reach into a file, take out facts, use them, and leave the file in perfect order. Computer files are great space-savers in offices, and are stored on magnetic disks and tapes. See **disk, tape.**

**first-in-first-out** (fûrst in fûrst oût), **FIFO**: a rule computers sometimes follow, when they pick the next number to add, subtract, etc. They choose the number that has been there longer than others (the "first one in"). It's like standing in line to buy movie tickets. The one who's there first gets tickets first. Computers don't always line up their facts this way. See **last-in-first-out.**

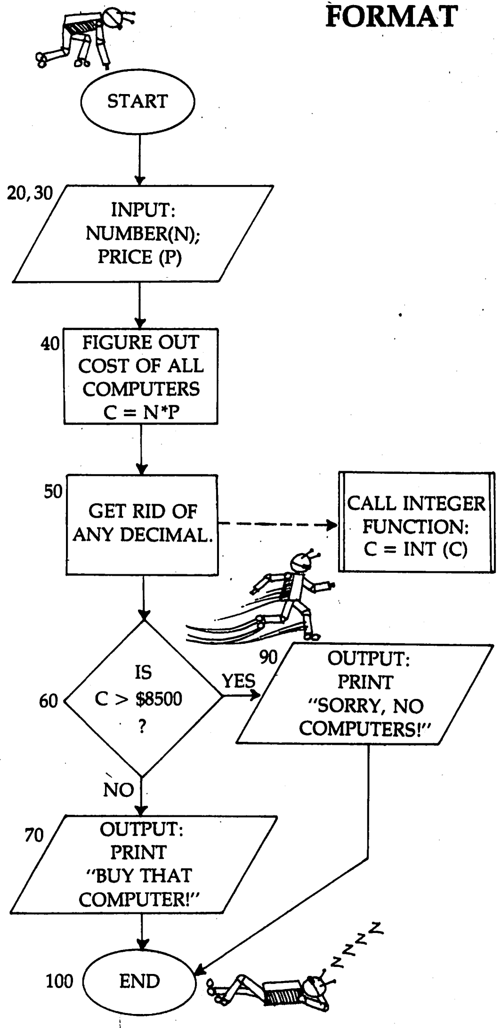
**flexible disk** (flek' sə bəl disk): see **disk.**

**floppy disk** (flop' ē disk): see **disk.**

**flowchart** (flō' chärt'): a plan of action for your computer; an outline of a program you are going to write. The purpose of a flowchart is to plan a good work flow for the computer and identify the steps you will have to write. In a flowchart, these steps go into special "boxes" that are symbols for program writers. Flowchart arrows show the movement from one step to the next. The chief flowchart boxes are (a) an oval-shaped box for "start" and "end"; (b) a rectangle for work like adding, multiplying, and comparing; (c) a rectangle, leaning over, for places where you give the computer some data, or it gives some to you; (d) a diamond-shaped box for big decisions that the computer makes; (e) a "double" sided box for a rule (a "function") the computer already has in its memory. The flowchart at the right was used to plan the program that follows. This program will find out if a school has enough money to buy a new microcomputer for each of its six (N) classrooms. The price (P) of one model is $1449.99 per computer. The school cannot pay more than $8500 total.

```
10 REM PROGRAM TO FIND COST (C) OF
COMPUTERS
20 LET N = 6
30 LET P = 1449.99
40 LET C = N*P
50 LET C = INT (C)
60 IF C > 8500 THEN 90
70 PRINT "BUY THAT COMPUTER!"
80 GOTO 100
90 PRINT "SORRY, NO COMPUTER!"
100 END
```

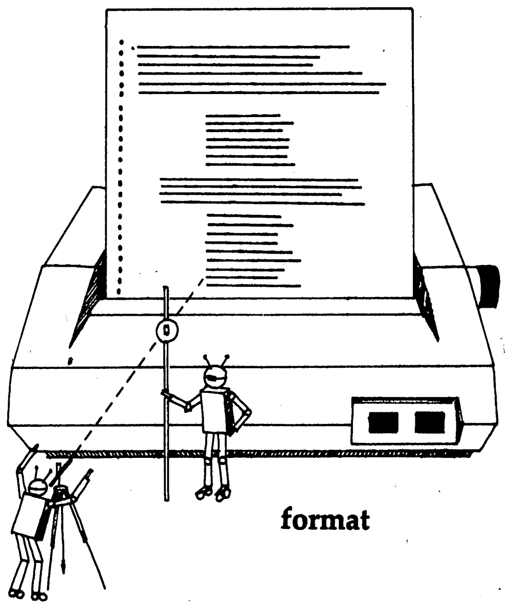
Too bad. No computers! See **program**.

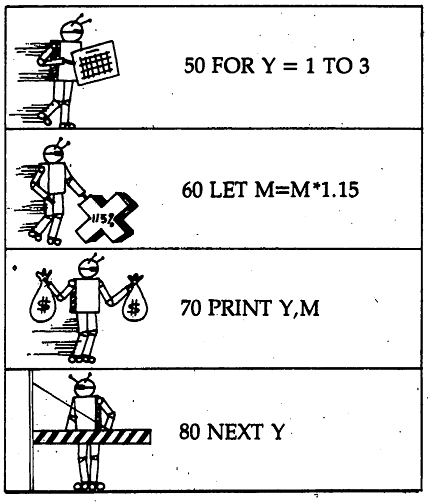

START
20, 30
INPUT: NUMBER(N); PRICE (P)
40
FIGURE OUT COST OF ALL COMPUTERS C = N*P
50
GET RID OF ANY DECIMAL.
CALL INTEGER FUNCTION: C = INT (C)
60
IS C > $8500 ?
YES
90
OUTPUT: PRINT "SORRY, NO COMPUTERS!"
NO
70
OUTPUT: PRINT "BUY THAT COMPUTER!"
100
END
**flowchart**

**FN-**: see **DEF FN-**.

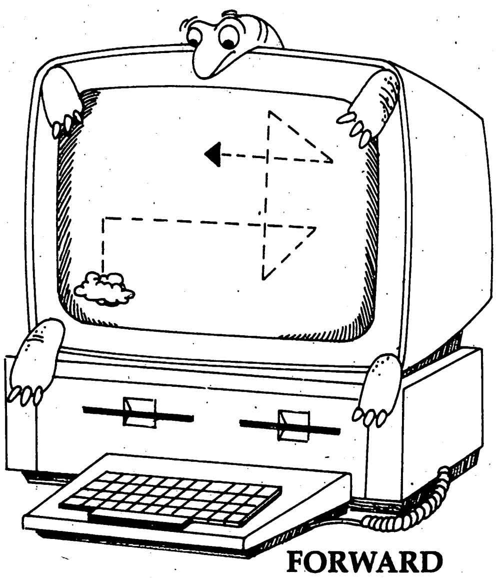
**format** (fôr′ mat′): **1.** a way of arranging material on a typed page. This includes setting margins on all sides, typing numbers in columns, and so on. In word processing, you have special typing keys that center a line or indent a paragraph. See **CENTER, INDENT, FORMAT**. **2.** a way of organizing data on a magnetic disk. The computer takes care of this format for you. It assigns a code number to each block of data you store on your disk. This code makes it possible for the computer to pick data out of the middle of a disk in a split second. **3.** sometimes, the arrangement of steps in a computer program. See **flowchart**.

**FORMAT**: in word processing, a key you touch to change margins, create spaces between lines, and arrange for columns of typing.

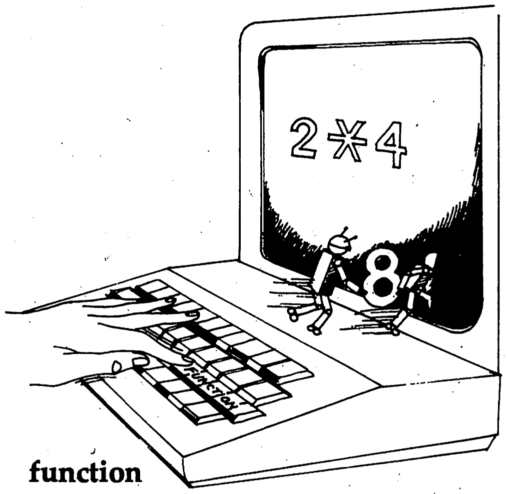

**format**



FOR . . . NEXT



FORWARD



function

**FOR . . . NEXT** (fôr nekst): in BASIC, a pair of commands that tells the computer how many times to repeat a set of steps. The FOR . . . NEXT commands are always used together, and they throw your computer into a "loop." The FOR command always starts the loop; the NEXT command ends it. An example? Suppose you want to invest $100 (M) in a project that will pay 15% interest every year (Y). You'd like to know how much you will have after three years. This FOR . . . NEXT loop in your program will get the answer.

```
50 FOR Y = 1 TO 3
60 LET M = M*1.15
70 PRINT Y, M
80 NEXT Y
```

Line 50 sets the count for your loop. It tells the computer to repeat your instructions each year (Y) for three years (1 TO 3). Line 60 holds the rule for working with your interest (M times 115%). It also keeps track of the value of M each year. Line 70 tells the computer to give you the value of M after interest has been figured for a given year. Line 80 acts as a loop checkpoint. If Y (the year) is not yet up to 3, the command NEXT Y sends the computer back to line 50 (to the "next" value of Y). As soon as Y is up to 3, the computer will break out of the loop, and go on to the following program step. Your investment value in three years? $152.09. See **loop, increment.**

**FORTRAN** (fôr' tran'), **FOR**mula **TRAN**slation: a "high-level" computer language used in science and mathematics. See **language.**

**FORWARD** (fôr' wərd), **FD**: in LOGO, a command that moves a small figure (a "turtle") on the computer screen. The command is followed by a number (for example, FORWARD 15). When you press the RETURN key, the turtle moves that many steps in the direction it is pointing toward. See **BACK, turtle, LOGO.**

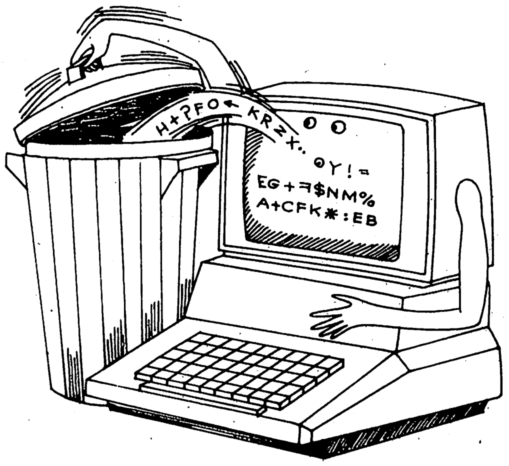
**function** (fungk' shən): **1.** in general, an action that is done, accomplished, carried out. *Adding is a computer function.* **2.** in a computer language, a rule that your computer can pull from its memory any time you ask. If your computer "knows" a rule when you buy it, you call the rule a "library" function. If you invent the rule and store it in your computer, it is a "user-defined" function. See **library function, user-defined function.** **3.** in word processing, extra help you can get by touching certain keys. See **COPY, MOVE, DELETE.**

## G

G: see **giga-**.

**games** (gām̄z): from Hangman to chess, one of the best ways to become friendly with a computer.

**garbage** (gär′ bij): **1.** data in the computer's memory that has not been used for a long time. Large computers have programs that test for garbage and get rid of it. **2.** data coming out of the computer that doesn't make sense. This kind of garbage often means there is a mistake in your instructions or in the data you're using. But the garbage can also be a sign that the computer itself is in trouble. In that case, the user's manual will suggest what to do. See **garbage in, garbage out; glitch; bug; test.**

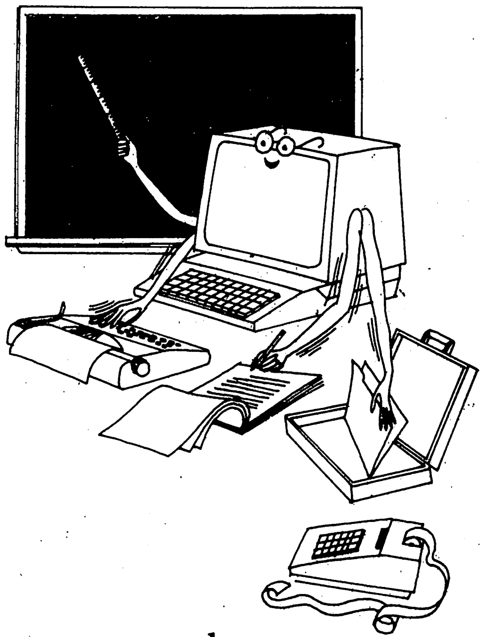


garbage

**garbage in, garbage out** (gär′ bij in′ gär′ bij oủt′), **GIGO**: a way of saying that your computer can't do a good job if you give it a sloppy program to work with. See **garbage.**

**gate** (gāt): see **logic gate.**

**general-purpose** (jen′ ər əl pûr′ pəs): able to handle many different tasks. A general-purpose computer is one that a writer, teacher, or businessperson might buy. See **dedicated**, for another kind of computer!

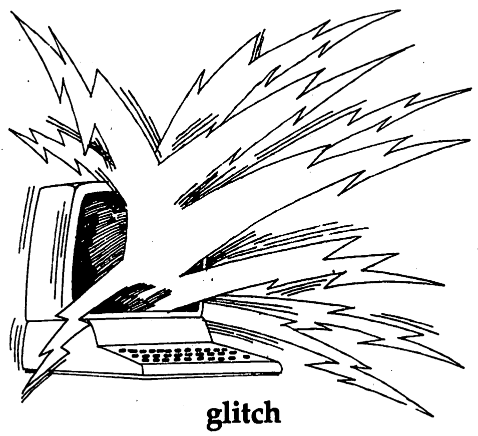

general-purpose

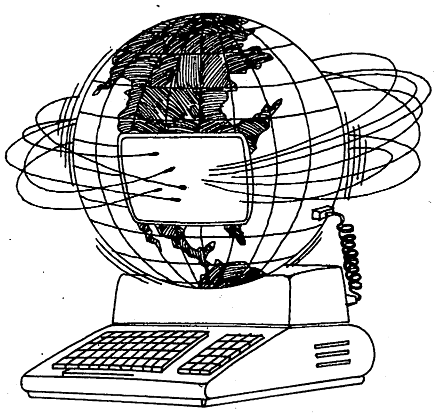
**general-purpose language** (jen′ ər əl pûr′ pəs lang′ gwij): a computer language that can be used for a variety of purposes. For example, PL/1, a computer language, is used in science and business. See **language.**

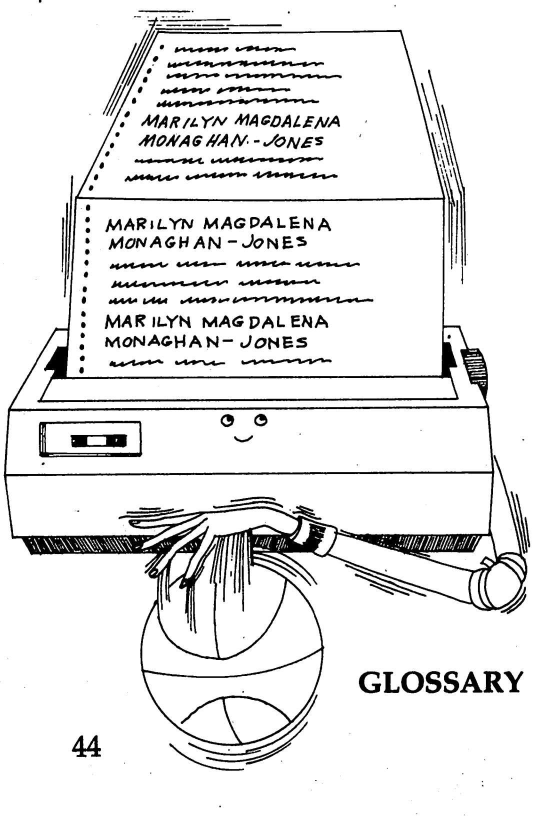
**ghost** (gōst): a light "double image" you may see on a page printed by a computer. Ghosts appear when the type "bounces" back against paper after printing on it.
**Can you see the ghosts in this line?**

**giga-** (jig′ ə), **G**: a prefix that changes a word to mean one billion times more. For example, giga- plus volt (a gigavolt), means one billion volts. That's a whole lot of electricity! See **gigabit.**

**gigabit** (jig′ ə bit′): 1,000,000,000 electric pulses; one billion "bits." Gigabit memory for small computers is a goal of computer research. Within a few years, your microprocessor will "remember" a billion bits of data, without the help of a disk or tape. That's roughly the same as 5,000 pages in a book like this. See **bit.**


glitch


global



GLOSSARY

**GIGO** (gī′ gō′): see **garbage in, garbage out.**

**GL:** see **GLOSSARY.**

**glitch** (glich): **1.** a sudden burst of electric energy in a computer. **2.** a blurring of the magnetic dots on a computer disk. Both kinds of glitch can cause a computer to lose its memory. (What dots?)

**global** (glō′ bəl): **1.** stored once in the computer's memory, but used over and over again. Suppose you write a program to keep records for the members (M) of your team. You would probably have a statement like this very early in your program:

```
    30 LET M = 15
```

Once you "define" M as 15, M will have that value throughout the program. In other words, M is an unchanging "global" term. **2.** covering the world. *Will we ever have a global computer system? (Do we need it?)*

**global hyphenation** (glō′ bəl hī′ fə nā′ shən): in word processing, your way of checking how words are broken at the end of a typed line. The computer shows you every word that has to be hyphenated (broken). You type the correct "break." Some computers have programs that hyphenate without your help.

**global REPLACE** (glō′ bəl ri plās′): in word processing, a way to change a word you typed in several places. Suppose you wrote a play about a girl named Teri. Now you want to change her name to Jennifer. With your REPLACE key, you get the computer to search for every "Teri" and change it to "Jennifer."

**GLOSSARY** (glôs′ ə rē), **GL**: in word processing, a key that tells the computer to find something in its memory, and insert it where you are typing. The GLOSSARY key saves you from typing the same name 20 times. Suppose you are typing a contract for a new basketball player whose name is Marilyn Magdalena Monaghan-Jones. You put Marilyn's name in your computer glossary. Then, every time you need it for the contract, you type GL plus a code letter (maybe M). The computer fills in her full name. If your team's name is the Uzey-Mozey Arkansas Basketball Brights, you might want to put that in your glossary, too. Code letter: U.

**GOODBYE** (gůd′ bī′): in LOGO, a way of erasing your work from the screen and from the computer's memory. Before typing GOODBYE, you can save your work on a computer disk. See **SAVE, disk.**

**GOSUB** (gō′ sub′): in BASIC, a command telling the computer to find a set of instructions which begins on another line. Here's an example of how it is used:

```
300 GOSUB 900
```

On line 300, you tell the computer to jump ahead to line 900. There, it will find a set of steps (sometimes called a subroutine), that it carries out right away. When it's finished, the computer goes back to line 310, and picks up where it left off! It may jump to line 900 several times during this program. GOSUB can save a lot of space in the computer's memory. See **RETURN, branch.**

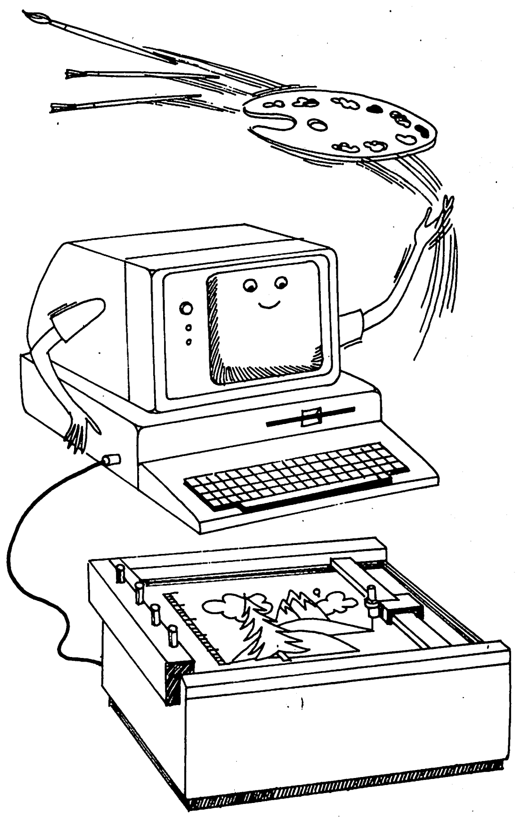
**GOTO** (gō′ tū′): in BASIC, your command to the computer to go to . . . any other line in your program instructions. It's called an "unconditional branch." See **branch.**

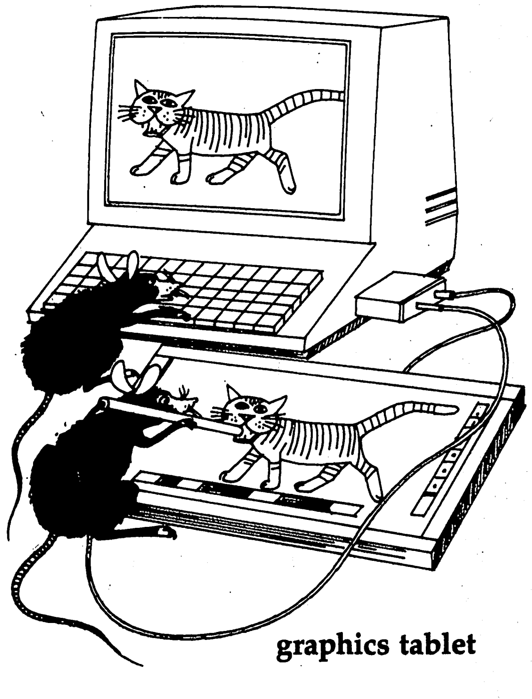
**graphics** (graf′ iks): graphs, diagrams, maps, and pictures that a computer shows on its screen. You can create graphics in several ways: (a) by typing instructions to tell your computer what parts of the screen to light up; (b) by drawing your picture on a graphics tablet (while the computer repeats it on the screen); (c) by outlining your picture on the screen with a "light pen." See **animation, graphics tablet, light pen, turtle.**

**graphics pad** (graf′ iks pad): see **graphics tablet.**

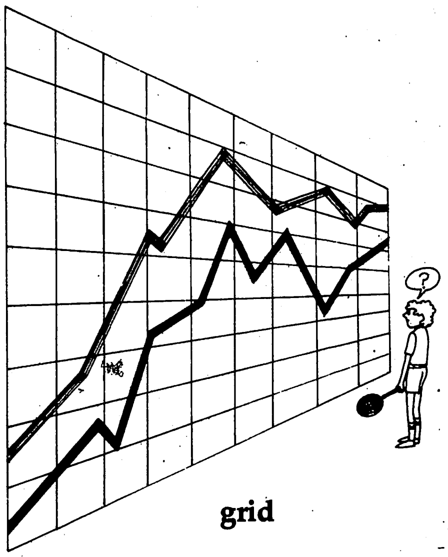
**graphics plotter** (graf′ iks plot′ ər): a computer attachment that lets the computer draw graphs, charts, and pictures right on paper. The plotter and its pen are attached to the computer. The computer moves the pen across paper in the plotter, and produces drawings from instructions in its memory. See **output.**

**graphics tablet** (graf′ iks tab′ lit): an attachment on which you draw pictures for the computer screen. You draw on the surface of the tablet with a special pen. As you press down, the tablet tells the computer where you are drawing. The computer then lights up matching parts of its own screen. The computer can store this drawing in its memory. See **graphics, input.**


graphics plotter

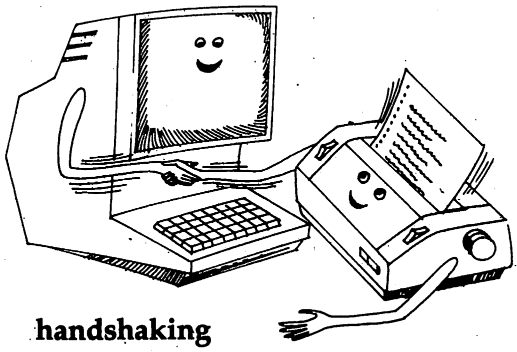

graphics tablet

**grid** (grid): **1.** top-to-bottom and side-to-side lines that cross one another. A piece of graph paper is a good example of a grid. **2.** a way of arranging information in columns, row by row. Seats at a concert form a grid. Each seat has two numbers: (a) the number of the row it's in; (b) its own number in that row (for example, row 6, seat 15). Your computer screen is a grid, too. It is divided into rows and columns—for example, 40 columns across, 25 rows down. A screen this size could display up to 1,000 characters (numbers, etc.) at a time.


grid

**gulp** (gulp): a term sometimes used for describing a handful of "nibbles." A nibble is a set of four electric pulses ("bits"). So a gulp might be eight or sixteen pulses. Computer scientists often go without meals when they work. Thus, the hungry words on their mind: **gulp, nibble, byte, menu, feed.** See for yourself.

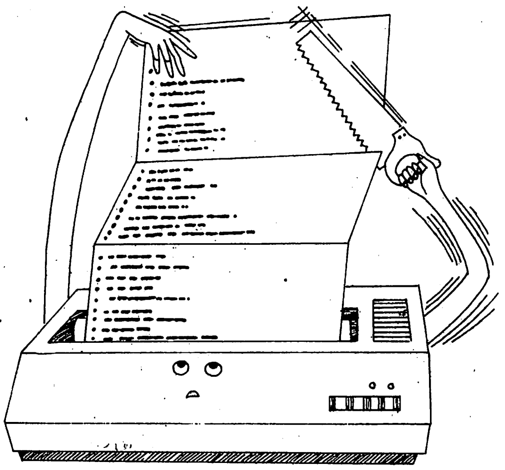
## H


handshaking

**handshaking** (hand' shā' king): a signal that one computer sends to another, before they exchange a message, to make sure the message will get through. Computers also "shake hands" with their attachments. "Hi, who are you?" "Your printer." "Who?" "YOUR PRINTER." "OK. Just making sure."

**hard copy** (härd' kop' ē): information printed on paper by a computer. Paper is not "hard," but we can keep it. Information on a computer screen is "soft" copy. That is, it disappears when the computer is turned off. Both hard and soft copy are forms of computer output. See **soft copy, output.**

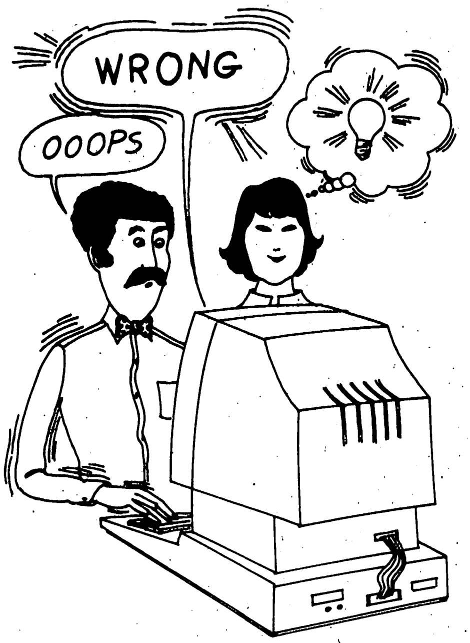
**hard disk** (härd disk): see **disk.**


hard copy

**hardware** (härd' wār): every part of a computer, except the programs you write for it or ask it to run. The hardware of a small computer includes a keyboard, screen, disk drive, and/or cassette player. Some small computers have a printer, too. Hardware also includes the computer's "insides"—its "chips," where all the work is done. Companies that make computer hardware include Apple, Atari, Commodore, Hewlett Packard, IBM, Osborne, Radio Shack, Texas Instruments, and others. See **software, computer.**

**head** (hed): see **read/write head.**

**heuristic** (hyū ris′ tik): solving a problem without the use of fixed rules. Heuristic solutions come from our guesses, experiments, and what we learn from mistakes. Heuristic thinking leads to inventions like the computer. It's also part of learning to get along with a friend. For problem-solving with rules, see **algorithm.**



**heuristic**

**hexadecimal system** (hek′ sə des′ ə məl sis′ təm): "hex," for short; a counting system that uses 16 numbers as its base (invented by someone with 16 fingers). Here are hex numbers, below the decimal numbers we use, and the binary numbers that computers work with:

| | |
|---|---|
| *binary* | 0, 1 |
| *decimal* | 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 |
| *hex* | 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F |

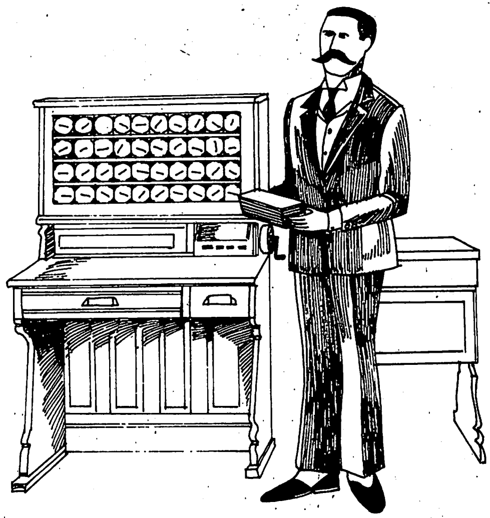
(The hex letters A to F stand for our numbers, 10 to 15.) Hex numbers (and letters) are helpful if you have to write a lot of 8-place binary numbers, like 00001111. You can use the hex form instead of binary. Here are the first 16 numbers in hex and binary:

| *hex* | *binary* | *hex* | *binary* | *hex* | *binary* | *hex* | *binary* |
|---|---|---|---|---|---|---|---|
| 0 | 0000 | 4 | 0100 | 8 | 1000 | C | 1100 |
| 1 | 0001 | 5 | 0101 | 9 | 1001 | D | 1101 |
| 2 | 0010 | 6 | 0110 | A | 1010 | E | 1110 |
| 3 | 0011 | 7 | 0111 | B | 1011 | F | 1111 |

How do you substitute hex for binary? Say you want to write 00001111. First, break that string in half (0000 and 1111). Then substitute the hex code for each half: 0 and F. Remember, that's not the word "OF," but two code symbols together. These 16 hex code units can be combined to represent every number, letter, and symbol on your keyboard. You have to check your manual to see if you can use the hex code on your computer. See **binary system, decimal system.**

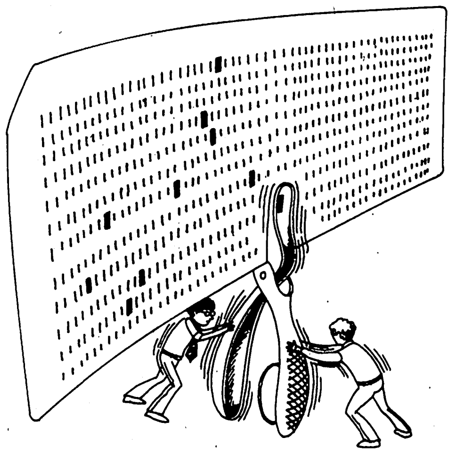
**high-level language** (hī′ lev′ əl lang′ gwij), **HLL**: see **language.**

**Hollerith, Herman** (hol′ ə rith hûr′ mən), 1860–1929: an American engineer who designed a punched-card system for taking the 1890 census. The census cards were counted electrically. This was the first time electricity was used in computing. See **Hollerith code.**



**Hollerith, Herman**

**Hollerith code** (hol′ ə rith kōd): a way of punching holes in cards to stand for numbers, letters, and symbols like the question mark. There are 80 columns and 12 rows in a Hollerith card. Computers "read" these holes, and store the information they contain. See **Hollerith, Herman,** and **input.**



Hollerith code

**human-oriented language** (hyū′ mən ôr′ ē ent əd lang′ gwij): a computer language that in some ways resembles everyday speech. BASIC and LOGO are human-oriented languages. See **language, BASIC, LOGO.**

**humans** (hyū′ mənz): **1.** people who make computers. **2.** people with car bumper-stickers saying, "Love a Computer Today" or "We Visited Computer Island." **3.** people who say, "Forget computers. I want my own satellite!"

## I

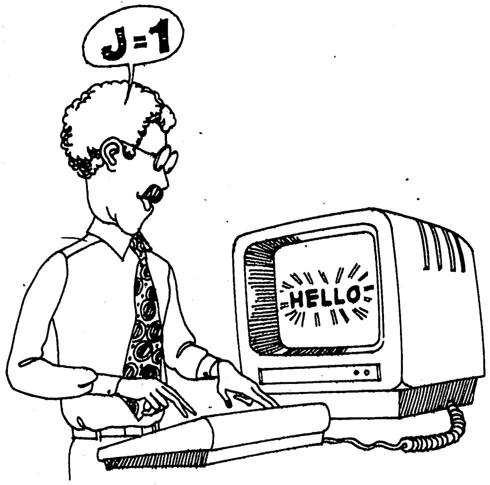
IC, Integrated Circuit: see **chip.**

**IFFALSE** (if fôls′): see **TEST.**

**IF . . . THEN** (if then): words telling the computer that it can skip steps in your program, provided a "test" is passed. "IF" sets up the test. "THEN" points out the line to jump to, if the test works. For example:

```
130 IF K = 12 THEN 150
140 GOTO 20
150 PRINT "THAT'S A DOZEN!"
```

In line 130, you tell the computer to check what's in its memory for K. (That's the test.) *If* the number stored in K is 12, *then* the computer will jump over line 140 to line 150. If the number is not 12, your computer slides right into step 140. See **branch** (definition 2).



IF . . . THEN . . . ELSE

**IF . . . THEN . . . ELSE** (if then els): in LOGO, words that set up a "test" and give the computer two ways to react to the results. Here's an example:

```
IF :J = 1 THEN STOP ELSE PRINT "HELLO
```

In this LOGO statement, you tell the computer to check what's in its memory for J. (That's the test.) If J does equal 1, the computer will then stop. If J does not equal 1, your computer will print a greeting. (That's the "else.") See **IF . . . THEN, branch.**

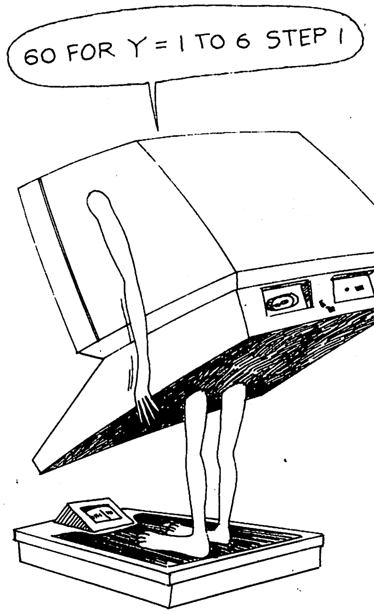
**IFTRUE** (if trū'): see **TEST**.

**impact printer** (im' pakt prin' tər): See **printer**.

**increment** (in' krə mənt): the amount by which a number is increased. If you say, "2, 4, 6, 8, 10, 12," etc., you're counting by increments of 2. You can get a computer to work by increments, too. Suppose you want to know how much you would weigh in six years, if you gained 2% of your weight each year. You could put this FOR . . . NEXT statement in your BASIC program:

```
60 FOR Y = 1 TO 6 STEP 1
```

Line 60 of your program tells your computer to repeat the calculation from the first to the sixth year (FOR Y = 1 TO 6). STEP 1 tells the computer to count from 1 to 6 by an increment of 1 (one). Your computer will take 2% of your "new" weight six times. See **FOR . . . NEXT**.



**increment**

**INDENT** (in dent'): in word processing, the key you touch to get a wider left-hand margin on your printed page (or on the screen). Sometimes you use INDENT if you are quoting words from another writer or speaker, like this:

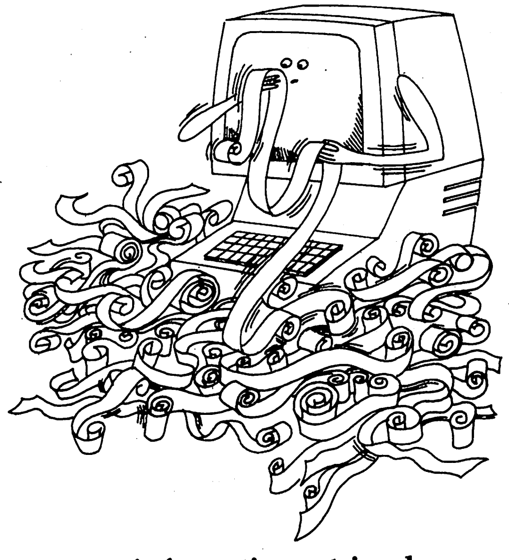
Our time is rich in inventive minds,
the inventions of which could facilitate
our lives considerably.

(Albert Einstein spoke these words in 1939.) See **TAB**.

**index** (in' deks): **1.** a list of items and their locations on a computer disk. If you have five poems on a disk, your computer "indexes" the name and location of each. **2.** the number for one item on a list. In a list of prices (P), the fifth price would be indexed as P (5). See **list**.

**information processing** (in' fər mā' shən pros' es' ing): see **data processing**.

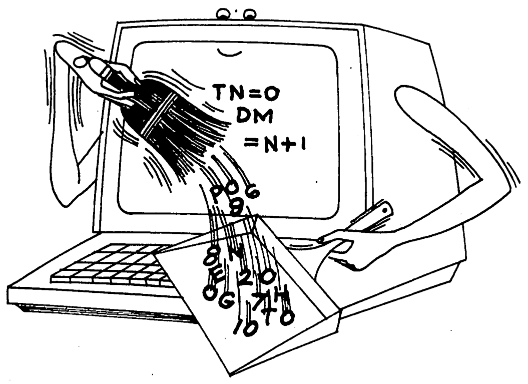
**information retrieval** (in' fər mā' shən ri trē' vəl), **IR**: finding a single fact in a great mass of data. Newspapers, large corporations, and the government look for millions of facts a day in their computer files. A good IR system depends on three things: (a) You need good labels for the files; (b) you must use these labels when you put new facts into the computer; (c) you must use these labels when you look up something.


**information retrieval**

**initialize** (in ish′ ə līz′): **1.** by a computer, to get ready to run a program; to clear a work area, for example, for instructions that will be coming in. **2.** with a magnetic disk, to prepare it to receive data. Your computer "initializes" a disk by checking it for defects. The computer also gives a code number to each section of the disk, to help find data later. See **address**. **3.** for a program writer, to set up a counting instruction. For example, suppose you want the computer to read the first seven marks (M) you got in English class. Here's part of your program in BASIC:

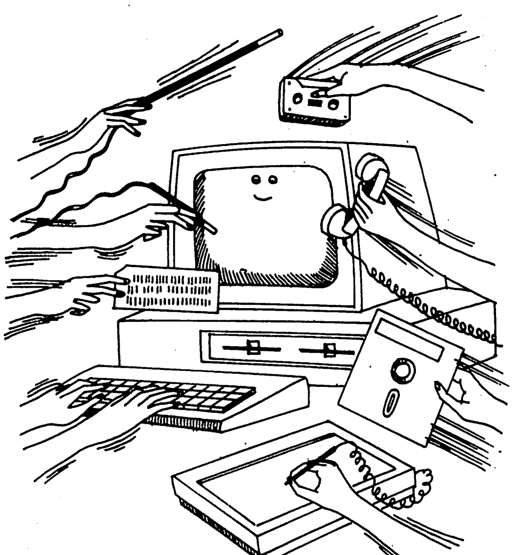
```
 90 LET N = 0
100 READ M
110 LET N = N + 1
120 IF N = 7 THEN 200
130 GOTO 100
```

Line 90 "initializes" the counter (N) at zero . . . before your computer reads any marks. Each time the computer reads a mark (line 100), line 110 increases the value of N by one. See **loop**.



initialize

**input** (in′ pŭt): a "movement" of data into the computer. You might tell a computer all it wants to know by (a) transferring data from a tape or disk; (b) having another computer call it on the phone; (c) typing instructions at a keyboard; (d) drawing on a "graphics tablet" attached to the computer; (e) using a "light pen" to "draw" on the computer screen; (f) waving a "wand" over the bar codes on store packages; (g) feeding hole-punched cards into the computer; and, one day, by talking to a computer that can recognize your voice. See **INPUT, disk, tape, modem, graphics tablet, light pen, wand, Hollerith code, output**.


input

**INPUT**: in BASIC, a word you use in your program to get the computer to ask you a question later (about a fact you must fill in). An example:

```
60 PRINT "WHAT WAS YOUR SCORE?"
70 INPUT S
```

When the computer "runs" your program, it will print the question in line 60. When it comes to INPUT S in line 70, it will stop, show a question mark, and wait:

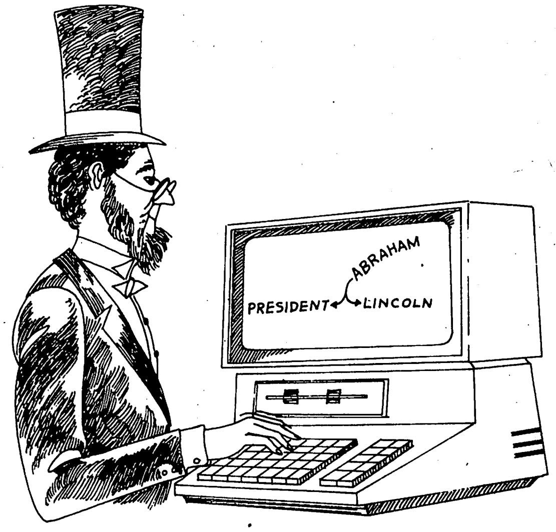
```
WHAT WAS YOUR SCORE?
?
```

That last question mark is for you. When you see it, you will fill in your score, and the computer will go on to the next step. See **input**.

**input/output** (in′ pût out′ pût) **I/O, IO**: a description of anything that helps get data into and out of a computer. An input/output "buffer" holds data that is waiting to be sent to or from a cassette. See **input, output, buffer.**

**insert** (in sûrt′): in BASIC, to add a new program instruction between two steps you have already typed. Suppose you write a 15-step program from lines 30 to 170. Later you remember that you left out a step between lines 40 and 50. You add line 45 after line 170. When your computer runs your program for you, it will "insert" line 45 right after line 40. See **line number, INSERT.**

**INSERT** (in′ sərt): in word processing, a key you touch to add a new word between two words you've already typed. The INSERT key helps when you revise (edit) a composition you are writing on your computer. Suppose you begin a story with the words "President Lincoln." Later, you decide to include Lincoln's first name. With the INSERT key, you tell the computer to put "Abraham" between "President" and "Lincoln." You can insert as many words as you wish. See **insert, edit, DELETE.**



INSERT

**instruction** (in struk′ shən): any step you want the computer to carry out. An instruction must name (a) the action you want the computer to take and (b) the data it should act on. Here's an instruction in BASIC:

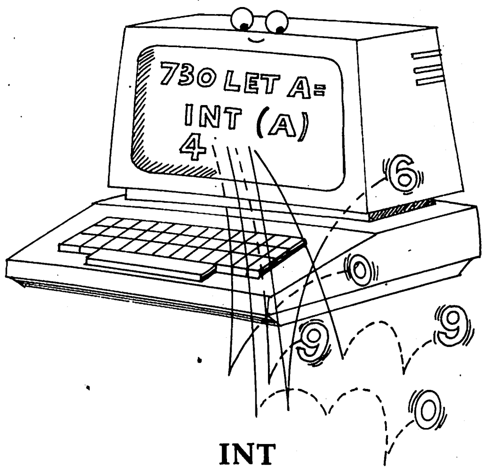
```
20 LET A = 3
```

You tell the computer to open up the memory space labeled "A." In it, you want the computer to store "3." In a "high-level language" such as BASIC, an instruction is called a statement. Means the same thing. See **statement, command, high-level language.**

**INT, INTeger function**: in BASIC, your instruction to the computer to drop the decimal part of a number. Suppose you give the computer a problem to work out, and you think the answer (A) will have a long decimal. Here's a way to get rid of the decimal:

```
730 LET A = INT(A)
```

If your answer (A) comes out as 4.99006, the computer will report it as 4. INT is a library function; a computer can do it without being told how. See **library function.**

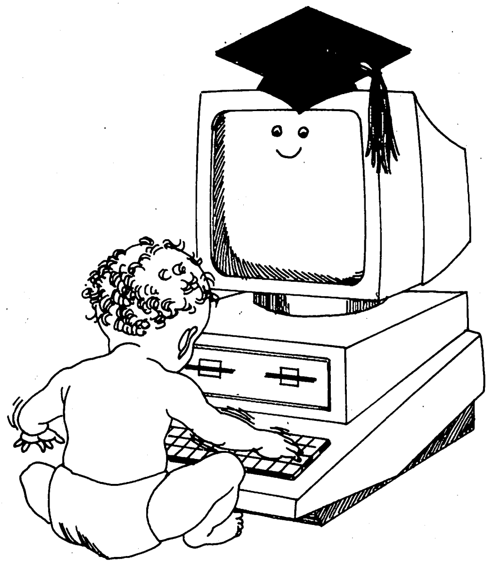


INT

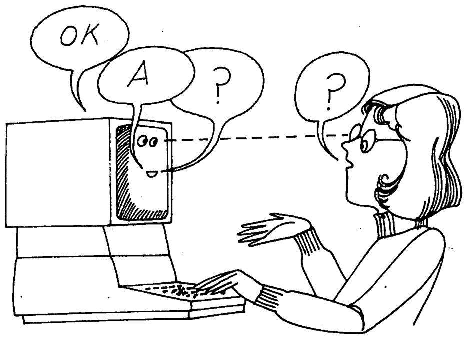
**integer** (in′ tə jər): a "whole" number; a number without a fraction or decimal. The numbers 1, 47, and 3,000 are integers, but 469.2 is not. Neither is 54½. See **INT.**

**integrated circuit** (in′ tə grāt′ əd sûr′ kit), **IC**: see **chip**.

**intelligent computer** (in tel′ ə jənt kəm pyū′tər): able to take new instructions, remember them, and carry them out. Both your computer and your washing machine carry out instructions. But your computer is called "intelligent"; your washing machine is not. The difference? You can give your computer a new kind of task every day. But you cannot make up new instructions for your washing machine. See **unintelligent computer, artificial intelligence, robot**.


intelligent computer

**interactive** (in′ tər ak′ tiv): having a conversation; asking and answering questions. An interactive computer system allows you to (a) give data and instructions to the computer; (b) look at the results; (c) change what you did; or (d) ask a question. And the computer may ask questions of its own.



interactive

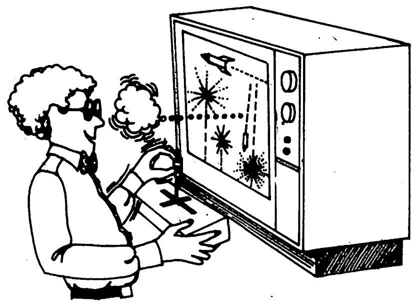
**interface** (in′ tər fās′): **1.** in general, the place where two units in a system meet. *The interface of Route 239 with Highway 630-A is at the edge of Ourtown.* **2.** a device that allows a computer to work with an attachment. A computer "speaks" in small electric pulses ("bits"). But a magnetic disk holds information in the form of magnetic "dots" that do not move. How does the central computer get those dots to talk to its bits? It has an interface that changes dots to bits (when the disk moves). In fact the work area of your computer needs interfaces for the keyboard, the TV screen, the printer, and other attachments. A common interface for small computers is the "RS-232." See **modem**.

**I/O, IO**: see **input/output.**

**IR**: see **information retrieval.**

# J

**joy stick** (joi′ stik): a handle or lever that you press to give instructions to a computer. By moving the joy stick you cause a shape on the computer screen to move in the same direction. This shape might be the "cursor" (a small light) or a figure in a computer game. See **paddle**.


joy stick

**jump** (jump): see **branch**.

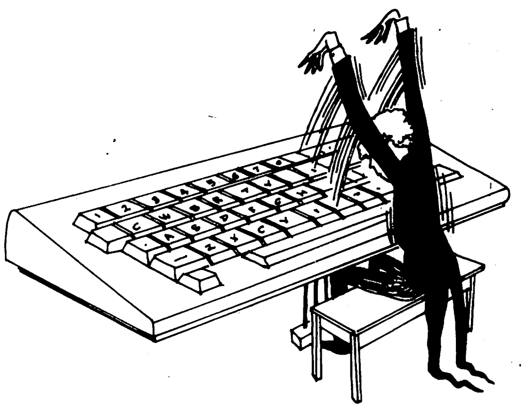
## K

**k**: see **kilo-** (definition 1).

**K**: see **kilobyte**.

**KB**: see **kilobyte**.

**keyboard** (kē′ bôrd): the part of your computer that looks like a typewriter. You use the keyboard to type instructions into the computer. Keyboards differ, but usually, you find all the standard typewriter keys, plus some "extras" such as a "BREAK" key. (This lets you interrupt anything the computer is doing). A few computers have keys for the "hex" code that some program writers use. See **input**, **hexadecimal system**.

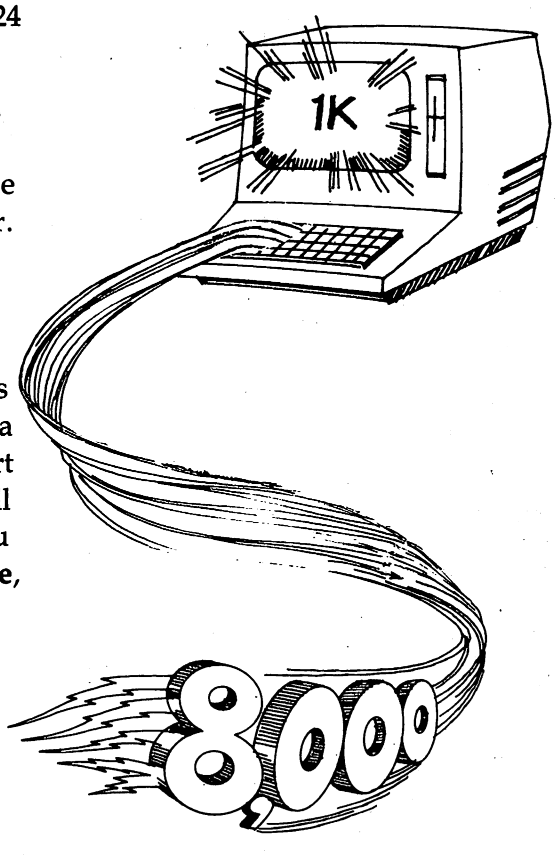

keyboard

**keypunch** (kē′ punch): a computer device that puts holes into computer cards or paper tape. The computer later "reads" a code from these holes. See **Hollerith code**.

**kilo-** (kē′ lō): **1.** usually, a prefix that changes a word to mean 1,000 times more. (The abbreviation is k.) For example, kilo- plus watt is a kilowatt, a thousand watts of electric energy. **2.** in computing, a prefix that changes a word to mean 1,024 times more. For example, a kilobit means 1,024 "bits" (electric pulses). See **kilobit**, **kilobyte**.

**kilobit** (kil′ ə bit′): a measure of how many electric pulses ("bits") a computer "reads." (Bits make up computer code.) One kilobit is equal to 1,024 pulses. It would take about one kilobit to put this definition into the computer. See **kilobyte**, **bit**.

**kilobyte** (kil′ ə bīt′), **K**, **KB**: a measure of how much information a computer can store. One kilobyte equals 1,024 "bytes." (A byte is usually equal to eight tiny pulses of electricity. These pulses are a computer's "code.") So, a kilobyte is roughly about 8,000 electric pulses. The short form for writing one kilobyte is 1K byte. Very often, you'll see just "1K." A 1K computer has very little memory. You could fill a 1K computer with half of this page. See **byte**, **bit**.



kilobyte

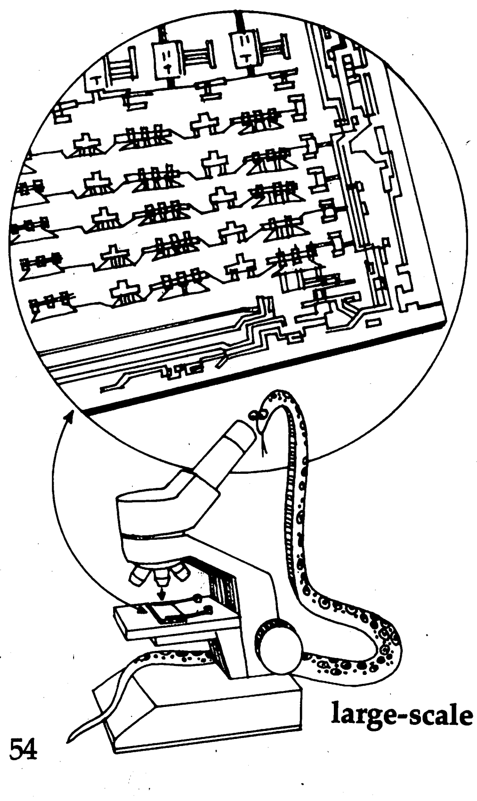
L

01010111
57
LET M= 2+3
W
W

language

**language** (lang' gwij): the letters, numbers, or other symbols you use for "talking" to a computer; the rules that tell you how to use these symbols. There are three kinds of computer language: "high-level" language, "low-level" language, and "machine" language. (a) With **machine language**, you write out instructions in "bit" code that computers understand directly. A bit code might need eight electric pulses for one letter of our alphabet. In some codes, for example, "W" is 01010111. Your computer carries out machine-language instructions instantly. (b) A **low-level language** uses a few letters or numbers for each set of eight bits. For example, "W" in low-level language can be coded as "57." The number 5 stands for 0101, and 7 stands for 0111.) Your computer has to translate a "low-level" instruction into a "machine" instruction before working on it, and has a special "interpreter" program for doing this! So it takes the computer more time to read a low-level language, but it takes you less time to write. Low-level languages are sometimes called "assembly languages." (c) A **high-level language** like BASIC or LOGO uses English words (or Russian words in Russia, French words in France). In BASIC you can write LET M = 2 + 3, just the way you "think" of that instruction. You save time writing your instruction in a "high-level" language and you don't have to learn any bit code. But your computer has a lot of work to do before it actually adds 2 plus 3. First, it has to decode "L," then "E," then "T," and so on. The computer is programmed to translate your high-level message into the bit code it understands. This takes your computer more time than reading its own machine language. What does "more time" mean in a computer's day? About a few millionths of a second!

**large-scale integration** (lärj' skāl' in' tə grā' shən), **LSI**: crowding more and more paths for electricity into a smaller and smaller space, specifically, on a tiny computer "chip." The more paths ("circuits") a chip has, the more electric pulses a computer can "read" at one time. It's something like a billboard with an electric sign: With just one light, the sign can only switch on and off. With 1,000 lights it can flash words. And with 10,000 (small) lights, it



large-scale integration

can switch different lights on and off to show pictures. People who design computers want as many "switch" points as possible, not, for flashing lights, but for handling the pulsing, electric code. And LSI designers keep on designing smaller and smaller circuits and switches to fit more of a computer's work on a tiny, quarter-inch chip. The goal in LSI today? One billion pulses ("bits") per chip. See **miniaturization, chip, circuit, transistor.**

**last-in-first-out** (last in fûrst oût), **LIFO**: a rule that computers sometimes follow when they look for the next number to add, subtract, or work with—they take the most recent fact in their memory (the "last one in"). It's like unpacking a suitcase. When you open it, the first item you take out is the last thing you packed. Computers don't always stack their facts this way. See **first-in-first-out.**

**least significant bit** (lēst sig nif′ i ˌkənt bit), **LSB**: in any group, the bit that is way over to the right. The zero in this string of bits is the least significant bit: 11111110. In this group, 01010101, the LSB is a "one." See **most significant bit, bit, byte.**

**LEFT** (left), **LT**: in LOGO, a command that turns a small figure (a "turtle") on the computer screen. Your command to turn LEFT must be followed by a number. LEFT 90 would make the turtle turn 90 degrees to the left. That's half of an "about-face" in the army. See **turtle, LOGO, RIGHT, FORWARD.**

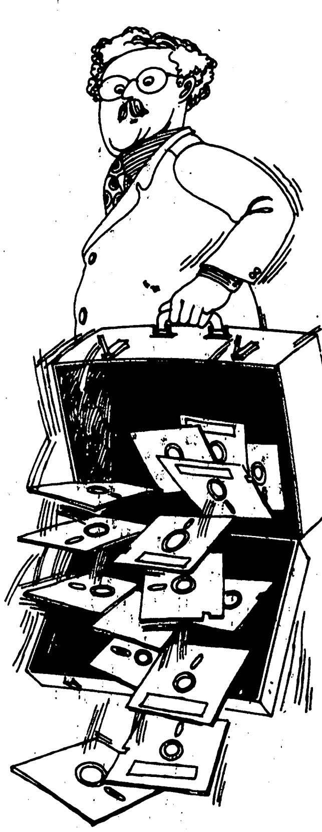
**LEFT$** (left string—the $ stands for "string"): in BASIC, a command to the computer to print part of a series (a "string") of items stored in its memory. The part that you want is on the left. Suppose you tell your computer the date of your birthday. You put it between quotation marks and give it a "string variable" name (R$):

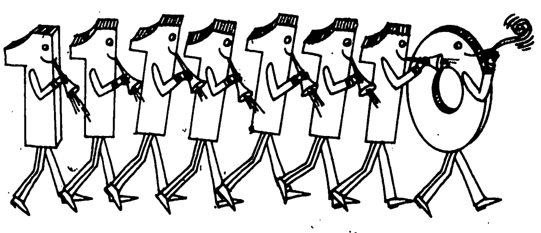
```
LET R$ = "OCTOBER 31"
```

When you want the computer to print it again, you type: PRINT R$. But if you want just the first three letters of the date, you type a print instruction like this.

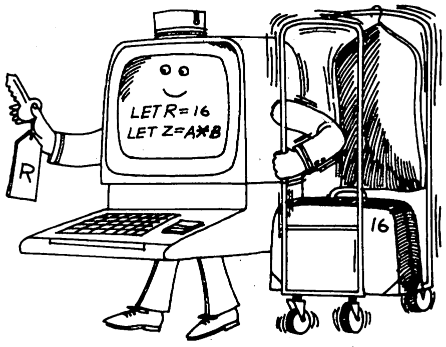
```
PRINT LEFT$ (R$, 3)
```

"LEFT$" tells the computer to read only the left part of R$. And the "3" tells the computer to print the first three letters. You can use the LEFT$ command for tasks like checking the alphabetical order of words. See **RIGHT$, MID$, string, string variable, list.**


last-in-first-out


least significant bit

**Leibniz, Wilhelm von** (lib′ nits vil′ helm von′), 1646 –1716: a German mathematician who invented a machine that would multiply, divide, and find square roots. His idea was great. His machine didn't work too well. But Leibniz got everybody else thinking about how it should work: (a) Break every problem down into its smallest steps, and (b) use a counting system with only two figures (0 and 1). Wilhelm had foreseen how computers think! See **bit.**



LET

**LET** (let): a BASIC command that feeds instructions and facts into the computer's memory. When you use LET in a program, you act like a hotel manager. You assign code names to "rooms" (places in the computer's memory). Then you tell the computer which "guest" (fact) may stay in each room. For example, here's how you could tell the computer to put the number "16" into one of its memory spaces called "R."

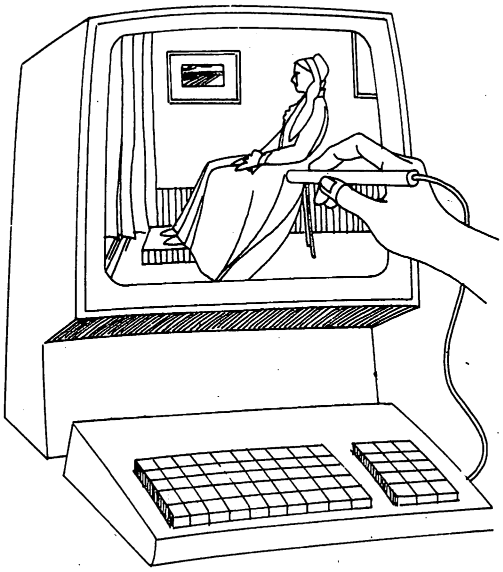
LET R = 16

Sometimes, you shuffle a few "guests" around. In the "LET" statement below, you tell the computer to find the facts in "rooms" A and B; then multiply them, and put the answer into "room" Z.

LET Z = A*B

LET lets you do a lot.

**library function** (lī′ brer′ ē fungk′ shən): a special task, usually in math, that your computer always knows how to do. To include a library function in your program, you use a code name which you will find in your computer manual. Suppose you want the computer to get rid of any decimal in a division answer, (A). In BASIC, with the code name INT, you would tell the computer: LET A = INT(A). Then if your answer has a decimal, 3.85572, your smart computer will change it to 3. Other library functions include finding square roots and selecting random numbers. See **user-defined function.**

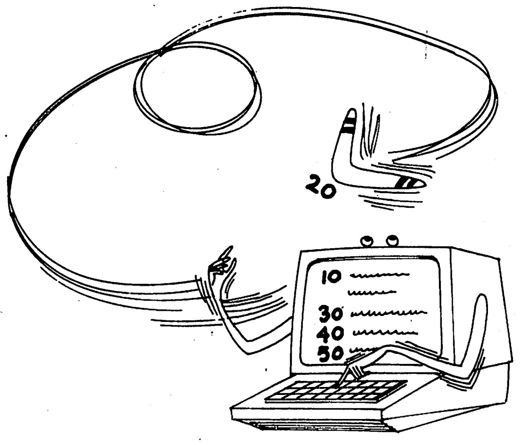

light pen

**library routine** (lī′ brer′ ē rū tēn′): see **library function.**

**LIFO** (lī′ fō): see **last-in-first-out.**

**light pen** (līt pen): a special pen that allows you to create or change a drawing on a computer screen. This pen is attached to the computer. When you move the pen to the screen, the computer "sees" it, and lights up the tiny area in front of the pen. As you move the pen, the computer "draws" your design. Light pens are also used for writing musical notes on computer screens. (Some computers play music.) See **graphics, synthesizer, computer-aided design**.

**line number** (līn num' bər): the number in front of an instruction line in a computer program. As you write a program in BASIC, you give a new number to each line. These numbers usually increase by 10s (10, 20, 30, etc.). Your computer carries out your instructions starting with the lowest number. To change an instruction that is already in the computer, simply retype its line number and type your new instruction. To drop an instruction, just retype its number and touch the RETURN key. In both cases, the first form of the instruction vanishes.



**line number**

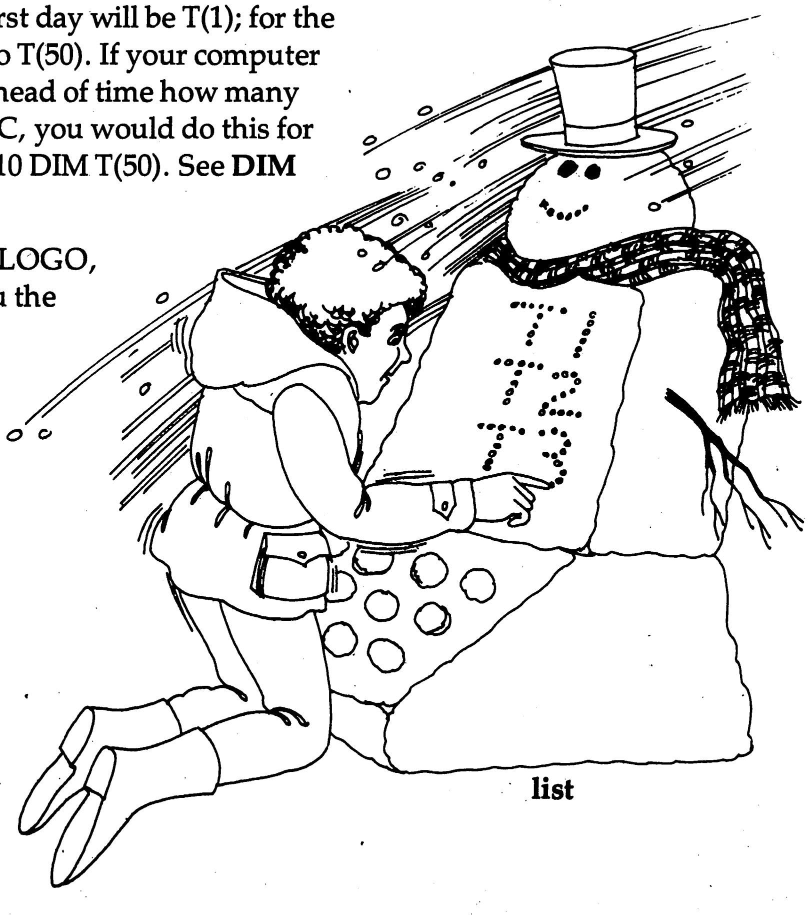
**line printer** (līn' prin' tər): see **printer**.

**LISP** (lisp), **LISt** Processing: a "high-level" computer language, used for working with lists of information. LISP is also used to help invent other computer languages. See **language**.

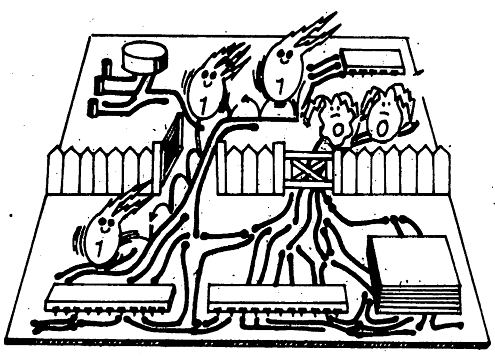
**list** (list): a series of related numbers; a string of data that the computer holds under one "name." This name is usually a letter of the alphabet. Suppose you want to record the temperature for the first 50 days of winter. You name the list: T. The temperature for the first day will be T(1); for the second day, T(2); and so on, up to T(50). If your computer is fussy, you may have to tell it ahead of time how many items your list will hold. In BASIC, you would do this for your temperature list by typing: 10 DIM T(50). See **DIM statement, string, array, LIST**.

**LIST**: a command in BASIC and in LOGO, telling the computer to show you the instructions in a program. Usually, the computer will show or "list" them on the screen. Using LIST is like reviewing your vacation plans in case you want to add or change a step. See **list**.

**LLL**, Low-Level Language: see **language**.


**list**

load


logic gate

**load** (lōd): **1.** to supply electric power to a computer. **2.** to get a computer attachment ready for action. Suppose you use a disk drive to transfer disk data into the computer's active memory. At the beginning, your computer loads the "head," the part that reads the magnetic disk. This brings the head and the disk surface together, something like turning on a stereo. See **disk, disk drive, read/write head. 3.** to put information (for example, a program) into the computer's working memory. See **LOAD.**

**LOAD:** in BASIC, your command to the computer to copy a program from magnetic disk or tape and show it on the screen. The LOAD command does not destroy the copy on your tape or disk. So for a while, your program is in two places in your computer. Here's how you might load your spelling program from a magnetic disk:

```
LOAD GENIUSINSPELLING
```

See **SAVE, disk, tape.**

**logic** (loj' ik): **1.** the ability to think, or "reason," correctly. If all coins are money, and a nickel is a coin, then a nickel is money. That's a nickel's worth of logic. **2.** in writing programs, the process of breaking down a problem to "yes-no" steps. You can't tell a computer, "Sometimes do line 50 and sometimes do line 60, instead." A computer doesn't understand "sometimes" but it does understand this in BASIC:

```
40 IF 5 <6 OR 10 = 11 THEN 60
50 GOTO 20
60 PRINT "ONE OF THESE IS TRUE"
```

The logic is in the "IF" statement. You tell the computer to test two statements. If one OR the other is true, then the computer goes to the instruction on line 60. See **logic operator, OR operator. 3.** the part of a computer that carries out a logic task; the "arithmetic and logic unit." See **arithmetic and logic unit, logic gate, circuit.**

**logic gate** (loj' ik gāt'): in a computer, any place where an electric pulse (a "bit") can be stopped or allowed to pass. Logic gates make it possible to show the result of a "logic" test. Suppose you tell the computer:

```
90 IF A = 54  AND B = 46  THEN 400
```

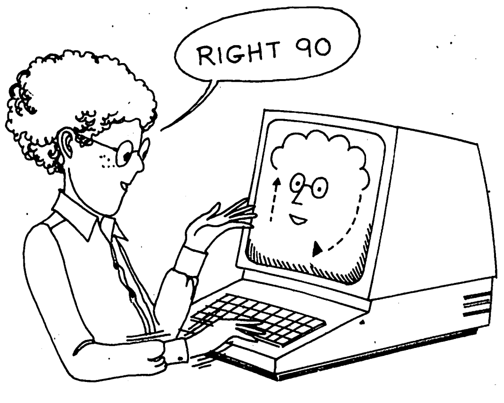
At instruction line 90, your computer will test the values of A and B right away. If its memory tells it that A is 54 *and* B is 46, your computer will go to line 400 ("THEN 400").

What happens if A is only 53? The AND "gate" stays closed, and electric pulses race instead to line 100 (or whatever number your next instruction is). See **logic operator, circuit.**

**logic operator** (loj' ik op' ə rā' tər): any word that tells the computer to make and compare two tests before its next move. Logic operators in BASIC are familiar words: AND, NOT, and OR. Each computer language sets up a logic test in its own way. In this LOGO statement, the logic operator is ANYOF:

```
    PRINT ANYOF (5< 6) (10 = 11)
    TRUE
```

In the first line, you want the computer to look at two comparisons. Is it true that 5 is less than 6? Yes. Is it true that 10 equals 11? No. Why does the computer print TRUE, when one comparison is false? Because ANYOF means: "find just one true comparison." We use logic operators all the time. Here's an example: "If Mom comes home early, *or* if my sister will babysit, I can go to the game with you." Either one will do it. Logic operators are sometimes called relational operators, but there is a difference. See **logic, AND operator, OR operator, relational operator.**

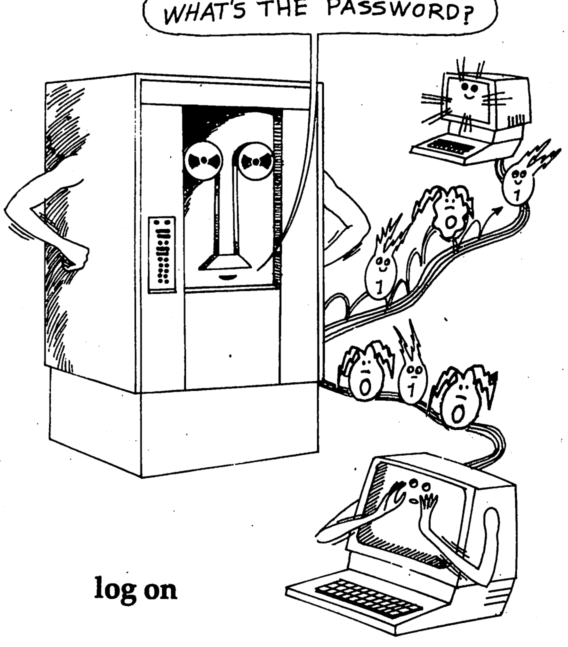


LOGO

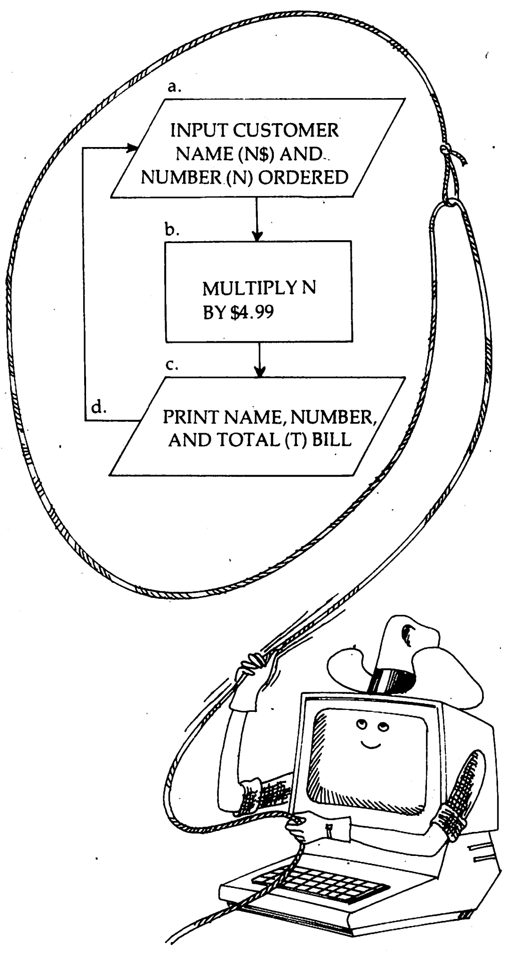
**LOGO** (lō' gō): a "high-level" computer language that helps students to write their own programs. One of LOGO's features is a "turtle" that you can "teach" to draw circles and other shapes. See **language.**

**log off** (log ôf): to stop the flow of information between a central computer and your terminal (keyboard). To log off a computer is like saying "good-bye" on the phone, and it's important when you work on a middle-size or large computer. When you log off, the computer notes the date and the amount of time you just spent at your terminal. See **BYE, log on, time sharing.**

**log on** (log on): to let a central computer know that you want to begin work at your terminal (keyboard). You do this when you work at a large computer that other people use also. To log on, you type an I.D. number and a computer password. This step takes care of two needs: (a) It protects the privacy of any work stored in computer files. (b) It keeps track of the time each person spends on computer work. See **log off, time sharing.**

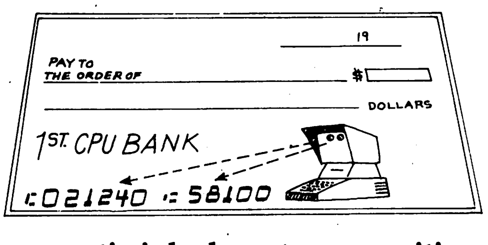


log on


a.
INPUT CUSTOMER NAME (N$) AND NUMBER (N) ORDERED
b.
MULTIPLY N BY $4.99
c.
d.
PRINT NAME, NUMBER, AND TOTAL (T) BILL

loop

```
10 REM BILLS FOR PET ROCK ORDERS
20 PRINT "CUSTOMER NAME, NUMBER ORDERED"
30 INPUT N$, N
40     IF N = 0 THEN 80
50     LET T = N*4.99
60     PRINT N$, N, T
70 GOTO 30
80 END
```



magnetic ink character recognition

**loop** (lūp): a set of instructions that you tell the computer to do over and over. A loop is like a set of exercises your coach shows you and then says, "Thirty times!" To set up a loop, you must tell your computer (a) what data (input) to use; (b) how to process the data; (c) what answer (output) to show; and (d) to repeat steps a, b, and c until you want it to stop. This kind of loop can be used for billing a lot of customers. Suppose you are selling pet rocks by mail, for example, and you want your computer to print a bill for each order. The flowchart at the left shows how the main steps of this job form a loop. Below the flowchart is one way to write a program for this loop in BASIC. (The loop instructions of a program are often indented so they are easier to read.) Notice that line 40 tells the computer to get out of the loop if there are no more orders. Programmers sometimes set up a "counter," an instruction telling the computer how many times to repeat steps in a loop. See **IF . . . THEN, FOR . . . NEXT, nested loop.**

**low-level language** (lō′ lev′ əl lang′ gwij), **LLL**: see **language.**

**LSB**: see **least significant bit.**

**LSI**: see **large-scale integration.**

**LT**: see **LEFT.**

## M

**m**: see **milli-.**

**M**: see **mega-.**

**machine code** (mə shēn′ kōd): see **language** (section a).

**machine language** (mə shēn′ lang′ gwij): see **language.**

**magnetic disk** (mag net′ ik disk): see **disk.**

**magnetic disk drive** (mag net′ ik disk drīv): see **disk drive.**

**magnetic ink character recognition** (mag net′ ik ingk′ kar′ ik tər rek′ əg nish′ ən), **MICR**: the way some computers read words printed in magnetic ink. Code numbers across the bottom of bank checks are printed in a special ink that has magnetic particles. An MICR computer is sensitive to each particle and records its exact location in a pattern of "dots." The computer then matches each pattern to a number (a character) in its memory. And it matches each set of numbers to a customer's name. Result? The bank knows it has your check. Banks with MICR computers can sort about 2,000 checks a minute. See **optical character recognition, optical mark recognition, wand.**

**magnetic tape** (mag net′ ik tāp′): see **tape.**

**mainframe** (mān′ frām′): **1.** the central work area of a large computer; the central processing unit of a large computer system, where the computer calculates and compares numbers. When a computer has several keyboard terminals, its mainframe is the most important section. **2.** sometimes, an entire large computer system is referred to as a "mainframe computer." It's like saying "wheels" when you mean a bicycle. The whole system is named by its most important part. See **computer, minicomputer, microcomputer, central processing unit.**

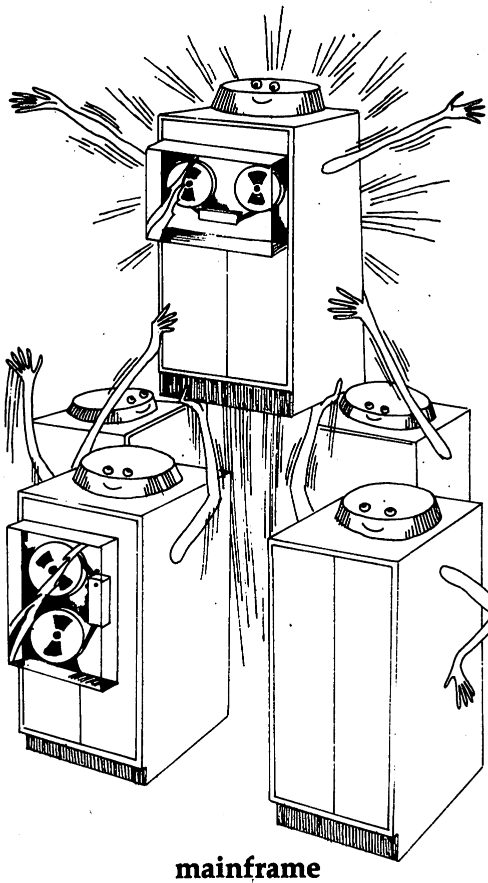
**manual** (man′ yū əl): see **documentation** (definition 2).

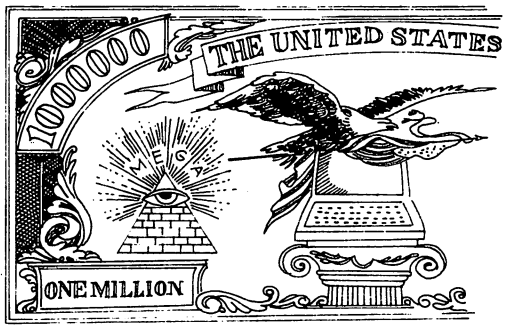
**MB**: see **megabyte.**

**M-byte** (em′ bīt′): see **megabyte.**

**mega-** (meg′ ə), **M**: a prefix that changes a word to mean one million times more. For example, mega- plus buck (megabuck) is slang for one million dollars. Megaton means a million tons—as in a nuclear bomb. See **megabit.**

**megabit** (meg′ ə bit′): a million "bits." A bit is a tiny electric pulse, the smallest piece of information a computer handles. There are two kinds of bit: a "one-bit" ("1" stands for an electric pulse that is "on") and a "zero-bit" ("0" stands for a very weak electric pulse, or a pulse that is "off"). A megabit is 1,000,000 of these pulses. Bits usually travel in groups of eight (for example, 01110110), and each group stands for a piece of computer code. A megabit, or 1,000,000 bits, sounds like a lot, but it wouldn't be enough to put this book into code. See **bit, byte, megabyte.**


**mainframe**

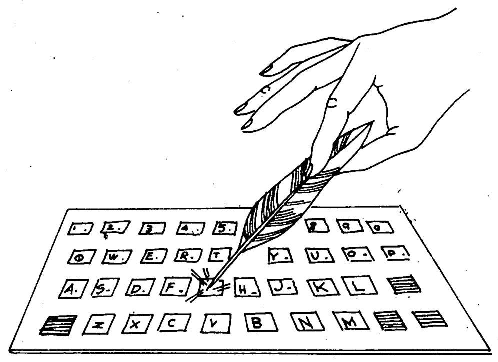


**mega-**

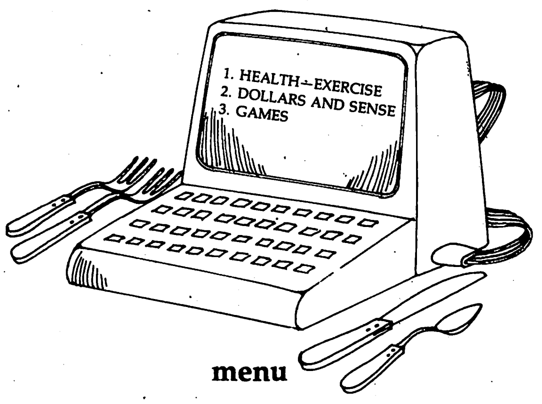
**megabyte** (meg′ ə bīt′), **MB, M-byte:** one million bytes; one million sets of electric pulses ("bits"). In some computers, a *byte* may contain as many as 32 bits, but the usual size is 8 bits. (This is a byte with 8 bits: 00110101.) A megabyte is one million of these. In computer codes, each byte represents a letter or number that you use. This page has room for about 2,500 letters or bytes. So a megabyte computer could hold 400 pages like this one in its memory. See **byte, bit.**

**member** (mem′ bər): see **tree.**

**membrane keyboard** (mem′ brān′ kē′ bôrd′): a piece of flat plastic with letters and/or numbers printed on its surface. It is used in place of typewriter keys for giving instructions to a computer. Just a light touch on any letter or number makes it work. Although a membrane keyboard may be less expensive, a keyboard with separate keys is better for fast, accurate work. See **keyboard.**


membrane keyboard

**memory** (mem′ ə rē): **1.** everything the computer knows, including the language it speaks, the facts you tell it, and the instructions it uses to do its work. There are two kinds of memory: (a) Permanent memory (ROM, or "read-only" memory) is in the computer when you buy it. It includes things a computer can always do, such as add. (b) Temporary memory (RAM or "random-access" memory) comes from your instructions to the computer while you use it. When you tell the computer to let A=3, this number is stored in RAM. It disappears as soon as you turn off the computer. To save what you're working on, you can transfer your work to a magnetic disk or tape before you turn off the computer. In a large computer system, disks and tapes are also called "memory." But with personal computers, disks and tapes are called backup storage. See **ROM, RAM, disk, tape, storage.** **2.** the places in a computer where its memory is located. For example, a ROM "chip" is the tiny, quarter-inch piece of silicon that holds instructions the computer always needs. See **chip, operating system.**



menu

**menu** (men′ yū): in a computer, the list of choices your computer shows you before you start work. Suppose your family keeps many computer programs on a disk. They are filed under different headings. When you get the computer working, the first thing you might see is a "menu" of these choices:

1. HEALTH—EXERCISE
2. DOLLARS AND SENSE
3. GAMES

If you want to check your running time, you could touch the "1" key (HEALTH—EXERCISE), which would get you another list (the second menu):

1. DAILY CALORIES
2. HEIGHT AND WEIGHT
3. RUNNING TIME

Now you would touch "3," and your program to check running time would immediately come up on the screen (much faster than using a restaurant menu.) See **prompt.**

**MERGE** (mərj): in word processing, a command that tells the computer to combine two lists, and print them as one. A school principal might MERGE the lists of both twelfth-year classes, to get one list of seniors.

**message** (mes' ij): **1.** in a large computer system, a whole chunk of data that is sent from one keyboard terminal to another. The FBI in Chicago, for example, might send a "message" of files by computer to the FBI in Atlanta. **2.** a communication from a central computer to one of its keyboard terminals. For example, the central FBI computer in Washington, DC, might send a message to its Chicago terminal: "Who told you to send those files?"

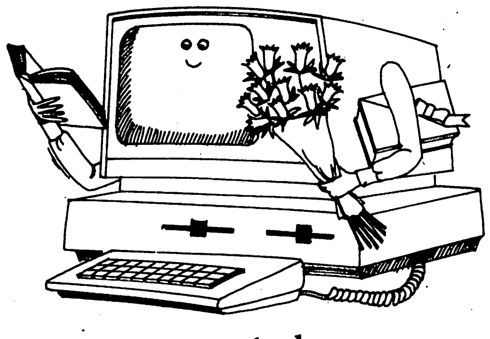
**metaphor** (met' ə fôr'): a way of describing one thing so that it reminds us of another. People often say that computers read, write, remember, and make decisions. These are metaphors, because they make us think of computers in human terms. Using metaphors is easier than saying, "electric current moved from one point to another, ten thousand times." Metaphors also make computers seem attractive to customers. See **user-friendly computer,** for example.

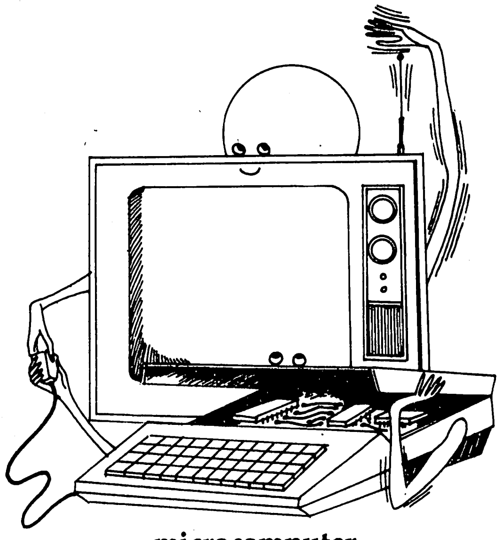
**MICR**: see **magnetic ink character recognition.**

**micro** (mī' krō): short form for microcomputer. *Our school just bought three micros for the science lab.* See **microcomputer.**

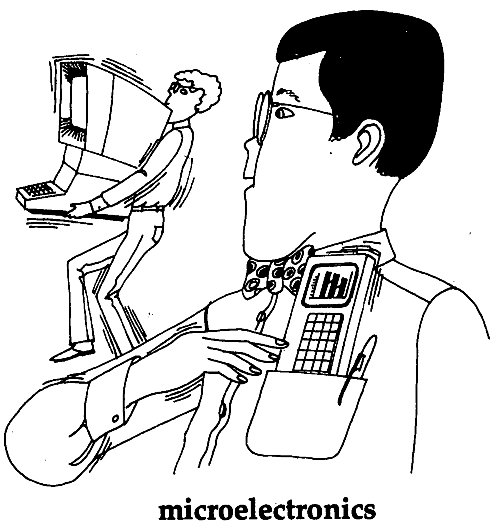
**micro-** (mī' krō): **1.** a prefix meaning "very small." Microfilm, for example, is film that stores photographs in very reduced size. See **microcomputer. 2.** a prefix meaning "one millionth of." A microsecond is one millionth of a second. Yes, there are clocks that can measure a million parts of a second! Now, see **nano-.**

**message**

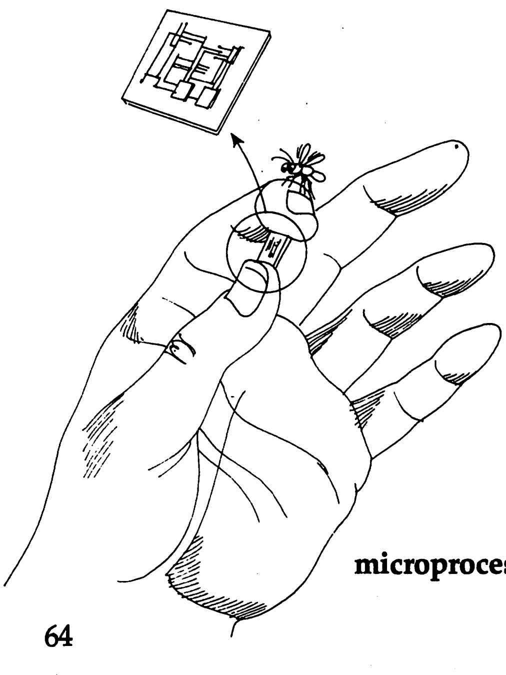

**metaphor**



microcomputer



microelectronics



microprocessor

**microcomputer** (mī' krō kəm pyū' tər): a "micro"; a "home," "personal," or "desk-top" computer; a computer smaller than a "mainframe" or "minicomputer." All micros have a keyboard, a microprocessor (its work area), and a screen—or a device for hooking up to your home TV. That's all your micro needs, to be able to take your instructions and work with them. But you can build a small system by adding extras: (a) cassette tapes and tape recorder for storing programs; (b) magnetic tapes, and disk drives, also for storing; (c) a joy stick or paddle for games; (d) a printer; (e) a modem, to connect your micro to the phone (so that it can "talk" to another computer); (f) a light pen or a graphic tablet for "drawing" on your computer screen. But there may be a catch: The computer you buy today may not work with the attachment you want in six months, or, you may have to buy an "interface" to connect the new device to your computer. So it pays to read a lot, ask a lot, try a lot before you buy even a little micro. It also pays to learn a little BASIC, the language most micros talk. See **computer, cathode-ray tube, keyboard, cassette, disk, tape, joy stick, printer, modem, light pen, graphics tablet, interface, compatibility, BASIC**. And see **mainframe, minicomputer**.

**microelectronics** (mī' krō i lek tron' iks): the science that produces computer "chips." If you look at a chip under a microscope, you will see thousands of electric paths (circuits). Each circuit can carry a separate piece of computer code. By increasing the number of paths, you can increase the power of a computer. Soon you'll be able to slip a powerful computer into your pocket, thanks to people who work in microelectronics. See **large-scale integration, transistor, circuit, semiconductor**.

**microprocessor** (mī' krō pros' es' ər), **MPU**: **1.** where all the action is in a small computer! The MPU is an entire central processing unit (CPU) or central work area located on a single tiny chip. (The chip is made of a square of silicon material, one-quarter inch on a side.) An MPU cannot handle the same amount of work as the CPU in a large computer. But it can take instructions, work on numbers, and remember answers. Of course, an MPU by itself is like a motor without a car. It needs a keyboard and other computer parts before it can work for you. Two popular microprocessors are the Z80 and the 6502. **2.** an MPU, plus extra chips and connections that together work as a personal computer. Sometimes the word "microprocessor" is used to mean microcomputer, but there is a difference. See **microcomputer, central processing unit.**

**microsecond** (mī' krə sek' ənd): one millionth of a second. That's just about how long it takes a fast computer to add two numbers.

**MID$** (mid' string—the $ symbol stands for "string"): in BASIC, a command to the computer to print part of a string of letters already in its memory. The part that you want is in the middle (MID$). Suppose you want to enter this date into the computer: 1986JAN21. You give it a "string variable" name (D$), and type it between quotation marks. If you want the computer to print just the month, but not the year or day, the second line below shows how MID$ helps you. (The last line is what your computer would print.)

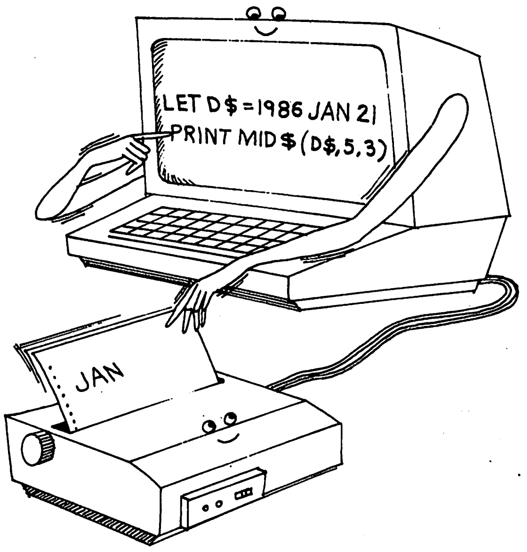
```
LET D$ = "1986JAN21"
PRINT MID$(D$, 5, 3)
JAN
```

"MID$" tells your computer to find the middle of D$. But where in the middle? Starting at the fifth place (5). And how much of the middle do you want? The next number to the instruction tells you (3). Now your computer will print JAN, the three "characters" you asked for. The MID$ command is the secret of how computers break down long strings of data into just the pieces they want. See **LEFT$, RIGHT$, string variable.**

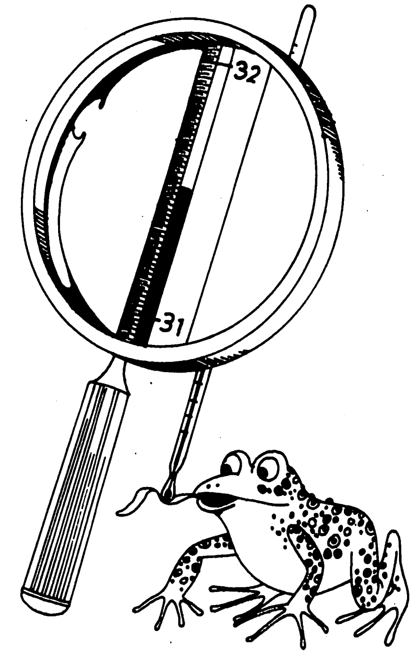
**milli-** (mil' ə), **m**: a prefix that changes a word to mean a thousandth part of something. For example, milli- plus degree (millidegree) means one thousandth of a degree. Shock your friends by telling them the temperature went up 50 millidegrees. (Only you could notice.) See **millisecond** and (hold on to your glasses) **millimicrosecond.**

**millimicrosecond** (mil' ə mī' krə sek' ənd): one billionth of a second. ONE BILLIONTH OF A SECOND! Blink your eye. You just lost a lot of millimicroseconds. Or, you lost a lot of nanoseconds. It's the same thing. See **nanosecond.**

**millisecond** (mil'ə sek' ənd): one thousandth of a second. Yawn! Someplace, an IBM computer just added 500,000 numbers in a millisecond. Now see **millimicrosecond.**

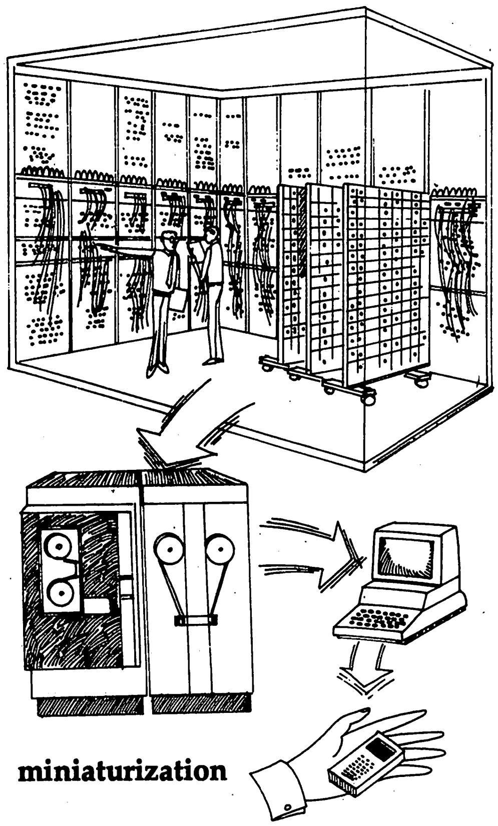


MID$



milli-

**miniaturization** (min′ ē ə chūr′ ə zā′ shən): making things in smaller sizes. It's the story of the shrinking computer. Thirty years ago, the first modern computer filled an entire room. Today, you can hold a computer with the same power in your hand. Tomorrow's computer size? Maybe the dot on this "i." See **large-scale integration.**


miniaturization

**minicomputer** (min′ ē kəm pyū′ tər): a "mini"; a "middle-size" computer; a computer that is smaller than a mainframe, but larger than a personal computer. The main work area (the "central processing unit") of a mini is in one cabinet. There are usually several keyboard terminals, and a separate storage cabinet for disks. A minicomputer can hold up to five megabytes in its working memory. That's about equal to 16 of these dictionaries. See **computer, mainframe, microcomputer, central processing unit.**

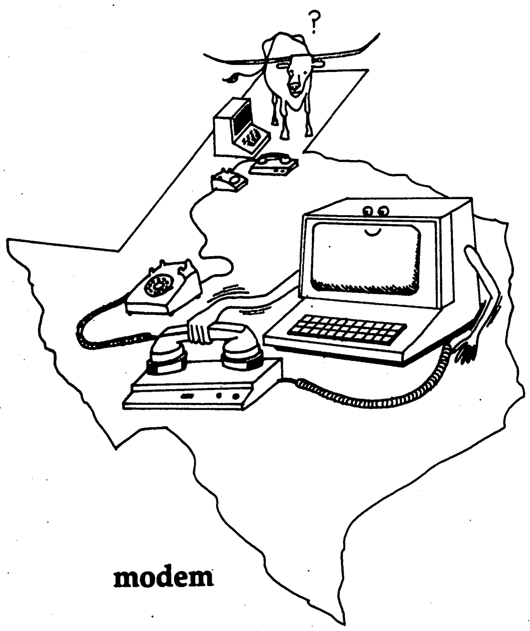
**minidisk** (min′ ē disk′): see **disk.**

**MIPS, Million Instructions Per Second:** what computer scientists want computers to carry out. To handle one million instructions in a second, a computer needs several work areas. See **central processing unit.**

**ML, Machine Language:** see **language.**

**mnemonic** (ni mon′ ik): something that aids your memory. Sometimes mnemonics are the key letters in a word we want to remember. Many computer terms are mnemonics: CTRL, for example, is a mnemonic for control key. READPICT is a LOGO mnemonic which tells the computer "read a picture in your memory, and show it on the screen."

**modem** (mō′ dəm), **modulator-demodulator:** a device that changes computer code into signals that can travel over a telephone wire, and vice-versa. You need a modem if you want your computer to "phone" another computer — to get news or other information from a data bank, for example. Many personal computers do not come with a modem. If you buy one, be sure it will work with your model computer. See **interface, data bank.**



modem

**modulator-demodulator** (moj′ ū lā′ tər dē′ moj′ ū lā′ tər): see **modem.**

**monitor** (mon′ ə ter): **1.** a kind of "self-check" that the computer performs for itself. The computer "monitors," for example, how the disk drive and the keyboard are working. It's like a patrol car, on a routine check. **2.** a TV screen at a terminal that receives information but cannot send it. Airports have monitors that show schedules of arriving and departing planes. **3.** sometimes, the screen on your personal computer. Unlike the use of monitor described in definition 2, you can actually give information to some personal computers by touching their screens with a "light pen." See **cathode-ray tube, light pen.**

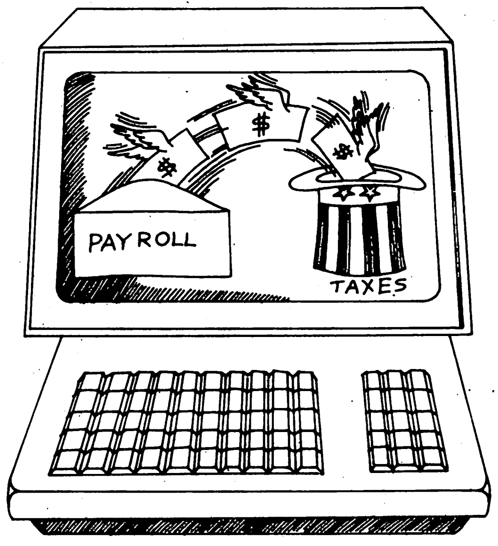
**most significant bit** (mōst′ sig nif′ i kənt bit′), **MSB**: in any group, the bit that is way over to the left. The zero in this string of bits is the most significant bit: 01111111. In this group, 10101010, the MSB is "one." See **least significant bit, bit, byte.**

**move** (mūv): generally, to "read" data in one place and "write" it in another. Computers often move information while they're working on a program. For example, a computer moves numbers from its payroll file to its work area to figure out weekly taxes. In this sense, the computer has the information in two places at once: the file and the work area. But see **MOVE** for a different meaning.

**MOVE**: in word processing, the key you use to switch words from one part of your text to another. The MOVE key is useful when you are editing a composition on your computer. Suppose you decide that the last sentence in your composition would look better at the beginning. You can just MOVE it back. But take a look at **move,** and see **COPY, INSERT, edit.**

**MPU**: see **microprocessor.**

**MSB**: see **most significant bit.**

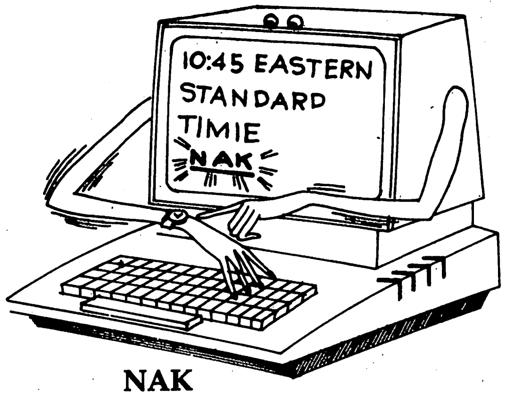


**move**

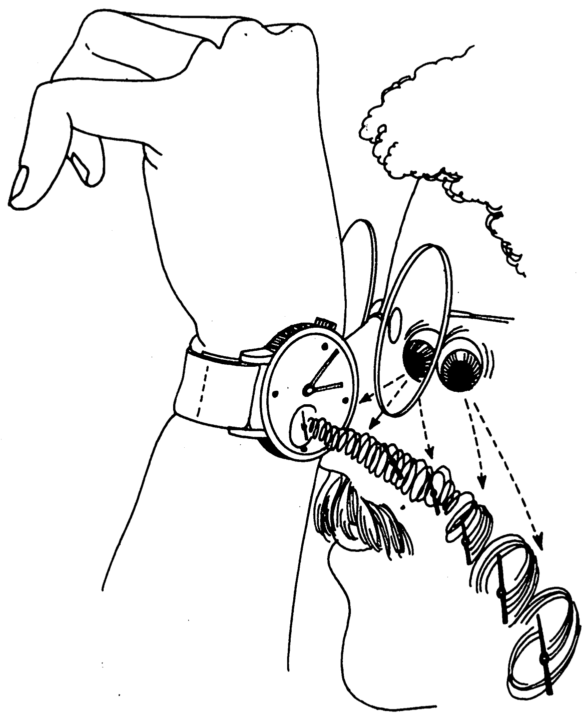
## N

**n**: **1.** see **nano-**. **2.** in mathematics, the symbol for "any number." For example, to explain how to count backward, you might say, "The rule is n − 1."

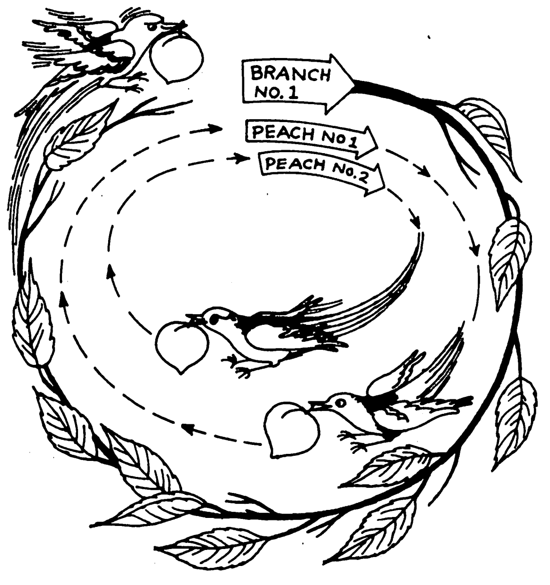
**NAK,** Negative AcKnowledge: a code meaning (a) the message you just received from another computer has an error in it; (b) you want the correct message to be sent. See **ACK, acknowledge.**



NAK



nanosecond



nested loop

**nano-** (nan′ ə), **n:** a prefix that changes a word to mean a billionth part of something. Nano- plus second (nanosecond) is one billionth of a second. That's something very hard to imagine. (So is a nanogoat.)

**nanosecond** (nan′ ə sek′ ənd), **NS**: one billionth of a second. This billionth-second has another name. See **millimicrosecond.**

NC: see **numerical control.**

**negative acknowledge** (neg′ ə tiv ak nol′ ij): see **NAK.**

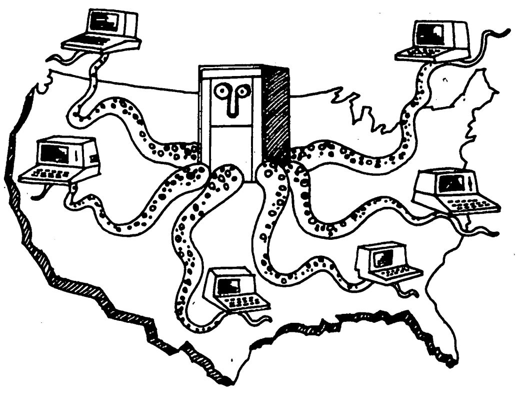
**negative number** (neg′ ə tiv num′ bər): in mathematics, a number less than zero. An example of a negative number is −15. See **positive number.**

**nested loop** (nes′ tid lūp′): a task the computer has to repeat in order to carry out another task it is repeating (like chewing between each swallow—for a person). Example from "real" life: Suppose you want to know how many peaches there are on each branch of your favorite tree. (Real life?) You would start with one branch, count all its peaches. Next you would go to branch number two, count its peaches; and so on. A diagram of these steps would show the second task (or loop) "nested" inside the first. Your "program" for this project would include steps like the following:

```
Program to Count Peaches on Each Branch
(a) Find favorite peach tree
(b) Find a branch
┌→(c) Give this branch a number
│   (d) Find a peach on this branch
│ ┌→(e) Give this peach a number
│ └─(f) Is there another peach?
│   (g) Remember the total of peaches on this branch
└─(h) Is there another branch?
(i) End
```

In the larger "loop" (c to h), you count off tree branches. In the "nested" loop (e to f), you count the peaches on a branch. Nested loop programs for computers are built the same way. A furniture store owner who needs to add up all the sales from each individual department, could use a nested-loop program on a computer. Your school principal might develop one to count all the absentees from each separate class. See **loop.**

**network** (net′ wərk): **1.** a system that includes one central computer and several (keyboard) terminals. These terminals might be in one room or in different cities across the country. People who use computer networks are on "time sharing." This means that ten or eleven terminals can ask for information at the same second, and get their answers one or two seconds later. (The computer gives a split second of attention to the first terminal, a split second to the next one, and so on.) Each person has the feeling that he or she has complete control of the main computer. See **time-sharing.** **2.** the way data is stored in a computer's memory. Each piece of data has its own "address." It's something like your telephone number: You have an area code, an "exchange" number (the first three figures), and your own private number (the last four). The computer organizes data in its memory the same way. See **address.** **3.** the organization of paths (circuits) on which electric pulses ("bits") travel. These bits are part of a computer code.


**network**

**NEW** (nū): in BASIC, a command that wipes away everything you just put into the computer's temporary memory. You use NEW if you make mistakes and want to start over. NEW does not change anything in the computer's permanent memory. See **memory.**

**newline** (nū' līn): see **NL.**

**NEXT** (nekst): see **FOR . . . NEXT.**

**nibble** (nib' əl): a series of four electric pulses; four "bits"; four of the smallest pieces of information a computer handles. Two examples: 0001, 1111. Eight bits form a "byte," a basic part of computer code. So, a nibble is also called a half-byte. No kidding. See **bit, byte, word.**

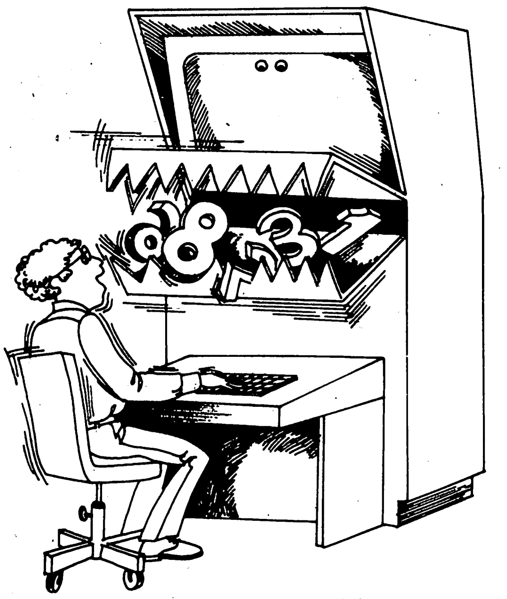
**NL, NewLine:** in some computers, the code for moving to the left margin on the next line. This can mean one of two things: (a) Your computer printer will move its paper one line ahead; or (b) a flashing light (a cursor) will move down one line, and to the left of the screen. See **cursor.**

**NODRAW** (nō' drô): in LOGO, your command to the computer to get ready for work that does not include any drawings. When do you use NODRAW? When you decide to switch from drawings to words and numbers. See **DRAW.**

**nibble**

**NS:** see **nanosecond.**

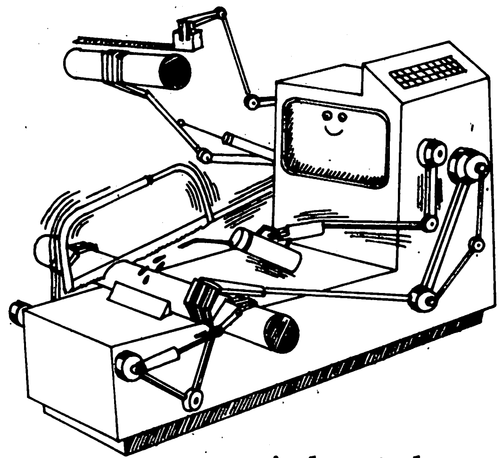
**number crunching** (num′ bər krun′ ching): the closest a computer comes to "Jaws." In one second, a large computer (mainframe) can chew up a million additions or subtractions. . . . And don't stand near it wearing a "times" sign (*).



number crunching

**numeric** (nū mer′ ik): made up of numbers, or digits (0 –9); not including letters of the alphabet. See **alphanumeric.**

**numerical control** (nū mer′ i kəl kən trōl′), NC: a way of organizing steps in a manufacturing task. Suppose pieces of metal have to be cut so that they are correct to a thousandth of an inch. The steps of holding the metal, measuring it, etc., can be numbered and coded by computers. The code is then fed into the cutting machine, which moves exactly the same way, at the same speed, to cut each piece of metal. See **computer-aided manufacturing.**



numerical control

## O

**OCR:** see **optical character recognition.**

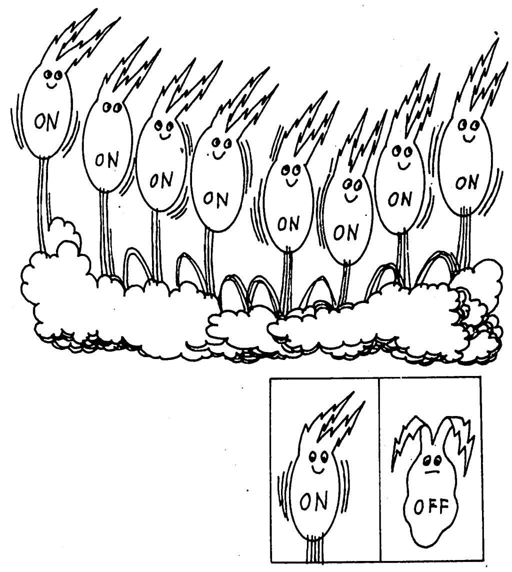
**off-line** (ôf′ līn′): at the moment, not controlled by the central computer. A printer is off-line when it is disconnected from the computer for repairs. See **on-line.**

**OMR:** see **optical mark recognition.**

**one-bit** (wun′ bit′), **1-bit:** a single electric pulse that the computer "reads," the same way we read our "ABCs." By itself, a single one-bit doesn't say much to a computer. (Neither does a zero-bit, which stands for "no pulse.") But a whole string of one-bits and zero-bits makes the computer sit up and "think." Usually, there are eight bits in a string (for example, 11000110, or 00000010). Few people learn to talk to their computers in one-bits and zero-bits. Most of us learn short-cut languages like BASIC and LOGO. When we type a BASIC command like LIST, the computer translates those four keystrokes into dozens of one-bits and zero-bits. Then it can put a list on your screen. See **bit, byte, binary, language, zero-bit.**

**on-line** (on′ līn′): currently connected to the main, or central, computer. Suppose a bank has a main computer in Kansas City. This computer is hooked up to terminals (keyboards) in 15 other cities. If your uncle is working at the Wichita terminal, he is "on-line" with the main computer. See **off-line.**

**on-off** (on ôf): **1.** a label for the switch you use to start a device and to shut it down. This switch has to be used carefully on a computer. Your computer manual will tell you "when" and "how." **2.** a description of every computer code. Electric pulses that go "on" and "off" form these codes. For example, a code for 255 is eight electric pulses: "on," "on," "on," "on," "on," "on," "on," "on." Yes, there's a short way to write that: 11111111. The "on" pulse is written as "1." The "off" pulse is written as "0." See **one-bit, zero-bit.**

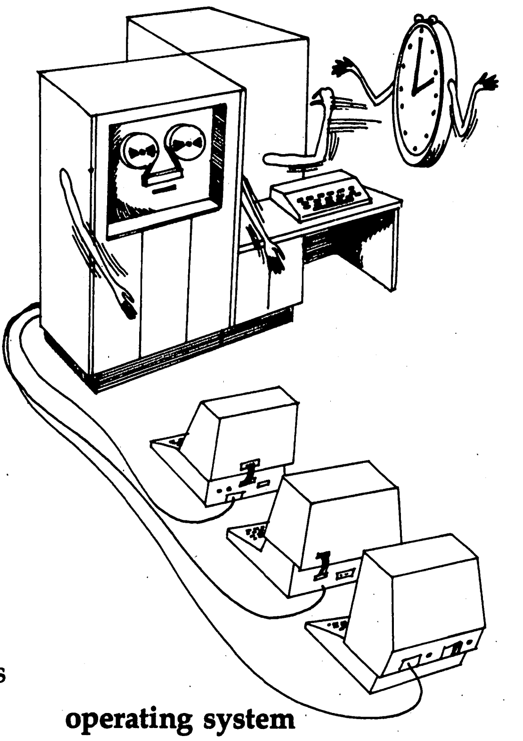


**on-off**

**operable** (op' ər ə bəl): able to be used; in operating condition. Your computer is not operable when the electric power goes out in a storm.

**operand** (op' ər ənd): **1.** any piece of information the computer works on; a piece of data that the computer adds, compares, etc. If you tell the computer: LET A = 10 + 9, you create two operands, 10 and 9, for the computer to work on. **2.** the pattern of "bits" (electric pulses) that the computer uses to find an operand in its memory. This bit pattern is like an address on an envelope. It has to match the "address" of the "memory-space" where the operand is stored. If "10" is in a memory space with the address 1111000011110000, the computer has to use 1111000011110000, when it wants 10. See **address.**

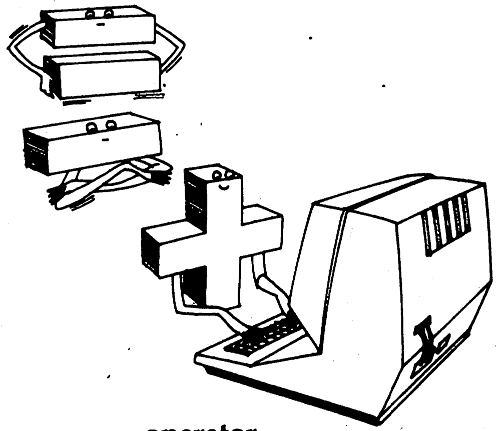
**operating system** (op' ə rā ting sis' təm), **OS:** instructions that are a permanent part of a computer. They tell the computer how to handle the programs you give it. In a middle-size or large computer, the operating system has other tasks, too: (a) It lines up all the jobs coming to the central computer from different terminals, and assigns memory space for each program. (b) The operating system gets the computer ready to work in different "languages" (BASIC, PASCAL, etc.). (c) The OS recognizes errors and helps to get rid of them. (d) It puts tapes and disks into action, when they are needed. (e) The OS also keeps track of the time that each terminal is on, and it makes sure that no one gets to use a program without a proper I.D. When the first big computers were built in the 1940s, a single human operator had to take care of all these tasks. The result? A huge computer would stand around waiting if the operator was trying to solve one problem. Today, OS systems check on everything that happens inside the computer. A popular operating system for small computers is the **Control Program for Microcomputers (CP/M).**


**operating system**

**operation** (op′ ər ā′ shən): **1.** any major task the computer takes care of. Printing is a computer operation. So is running Donkey-Kong. **2.** a single instruction that the computer carries out. In this sense, adding two numbers is an operation. See **operator.**

**operation symbols** (op′ ə rā′ shən sim′ bəlz): see **arithmetic operator, logic operator, relational operator.**

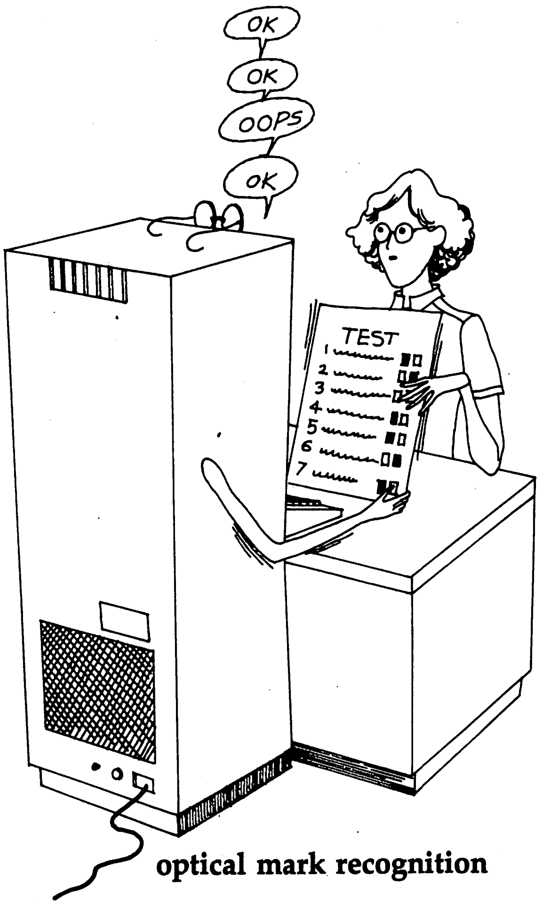
**operator** (op′ ə rā′ tər): **1.** a person who is running or using a computer. **2.** an instruction that tells the computer what to do with a particular piece of data. When you write operators, they are sometimes symbols, sometimes words. All operators tell the computer how to make one piece of data "work" with another: (a) *Arithmetic operators* (such as + or −) tell the computer to combine numbers. (b) *Relational operators* (such as the equal (=) sign) tell the computer to compare two things. (c) *Logic operators,* like the word "OR," tell the computer to test something, before taking one more step. See **arithmetic operator, relational operator, logic operator.**


operator

**OPM,** Operations Per Minute: the number of program instructions that a computer can handle in one minute. Usually, OPM means how fast the computer adds, divides, compares, and so on. See **operation** (definition 2).

**OPS,** Operations Per Second: the number of program instructions that a computer can handle in one second. Usually, OPS means how fast the computer divides, compares, adds, etc. See **operation** (definition 2).

**optical character recognition** (op′ ti kəl kar′ ik tər rek′ əg nish′ ən), **OCR:** the way some computers read printed pages, and even handwriting. When a computer with OCR looks at a page, it "sees" sets of dots. The computer tries to match these sets of dots with patterns it knows for our alphabet (A, B, C, a, b, c, etc.), and puts the whole "page" into its memory. (It's still not too good at handwriting. Handwriting samples are as different as fingerprints.) See **optical mark recognition, magnetic ink character recognition, wand.**



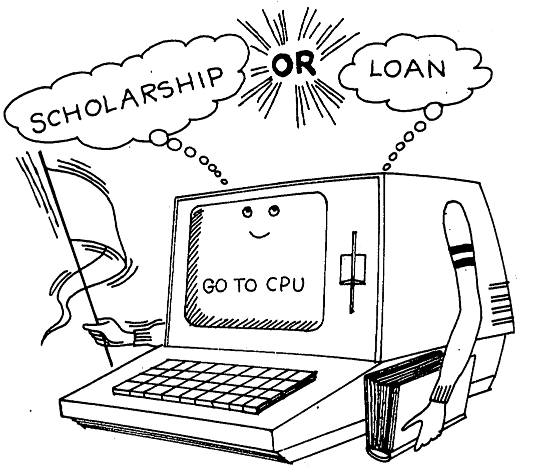
optical mark recognition

**optical mark recognition** (op′ ti kəl märk′ rek′ əg nish′ ən), **OMR:** the way some computers read the filled-in squares on tests. If you've ever filled in a circle or a square on a reading test, an OMR computer probably "read" your answers. How does it work? A computer finds a filled-in space on the answer sheet. It compares this mark with the pattern of "right" answers in its memory. Then it gives you a "plus" or "minus" score. You can't fool an OMR: Filling in all the choices for an answer will be marked "wrong," even though the "right" answer is one of them. See **optical character recognition, magnetic ink character recognition, wand.**

**OR operator** (ôr′ op′ ə rā′ tər): an instruction that gets the computer to check its memory before taking a special step. To instruct the computer when it should switch from one task to another, you might set up "conditions." If T equals 33, *or* if M is less than 100, for example, the computer can jump to a new step. In BASIC, you might set up the instructions with an OR operator this way:

```
400 IF T = 33 or M < 100 THEN 420
410 GOTO 250
420 PRINT "NEXT JOB"
```

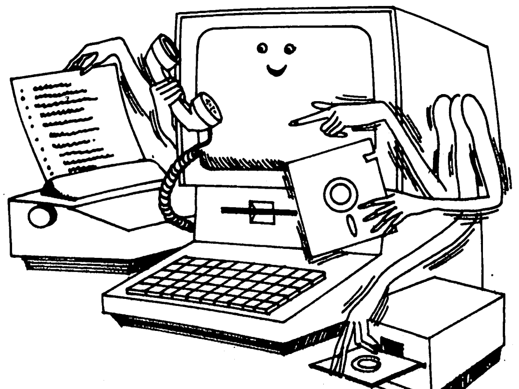
When your computer comes to the OR operator in line 400, it checks its memory. If it finds just one part of line 400 is true, your computer will jump to line 420 and print "Next Job." If not, it will follow the instruction in line 410. Using the OR operator is like saying, "If I win a scholarship *or* get a loan, I can go to college." See **logic operator, AND operator.**



**OR operator**

**OS**: see **operating system.**

**output** (out′ pu̇t): the "movement" of data *out* of the computer. There are many ways of getting output from computers: (a) Usually, your computer shows data on a screen (a "cathode-ray tube"). (b) With a printer, computers can give you pages of information on paper. (c) Using a "plotter," a computer will draw graphs and charts from data in its memory. (d) Computers transfer data to magnetic disks and tapes. (e) Using modems, they "talk" with other computers, sending information back and forth between them. (f) Some computers punch holes in cards as a way of storing coded data. (g) Computers can imitate speech (but not very well). (h) Some computers will pass you by, and put their output directly to work. For example, some computers tell robots what to do. See **cathode-ray tube, printer, plotter, speech synthesis, disk, tape, modem, robot, input.**



**output**

## P

p: see **pico-**.

**Pac-man** (pak′ man): that game; a "smile" button that eats quarters.

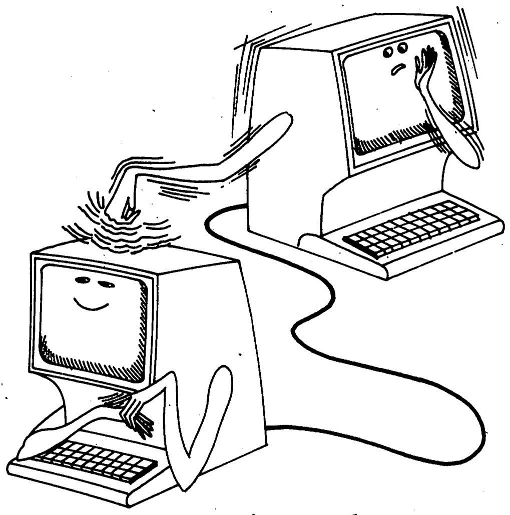
paperless office

**paddle** (pad′ l): a handle or lever that you press, to give instructions to the computer. As you move the paddle, a figure or shape moves in the same direction on your computer screen. A paddle can move in only four directions: forward, back, left, and right. But people often use "paddle" when they mean "joy stick." (A "joy stick" can move in any direction.)

**paperless office** (pā′ pər lis′ ô′ fis): a way of describing the "office of the future." Computers keep files, send mail, make out checks, order ball-point pens, and call customers. So why would anyone need paper? (To use the ball-point pens?)

**Pascal, Blaise** (pas kal′ blāz), 1623–1662: a French mathematician who invented a machine that could add and subtract. Pascal was 19 years old when he built this machine for his father's business. The business disappeared, but the family name lives on. See **PASCAL**.

**PASCAL**: a "high-level" computer language, named after a French scientist who lived centuries ago. See **Pascal, language**.


passive mode

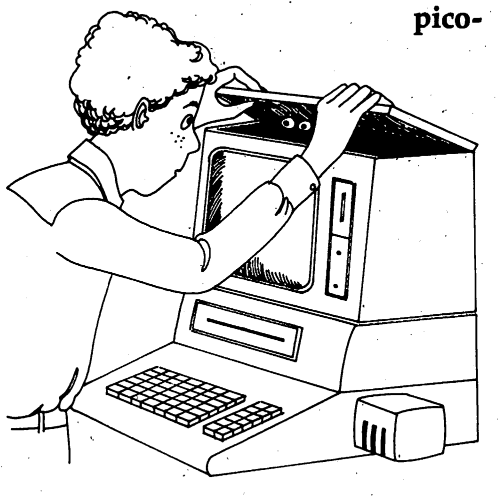
**passive mode** (pas′ iv mōd): **1.** not sending a message and not receiving one, either. A keyboard terminal is in the passive mode when it does not exchange messages with its main computer. **2.** able to receive data from the main computer, but not able to send data back. For example, a doctor might study a patient's chart on a computer screen, but he might not be able to add anything to it.

**password** (pas′ wərd): a code or piece of information you put in, to use a large computer system. Schools, hospitals, and companies try to protect their computer records by inventing passwords for people who read them. If you don't give the correct password, the computer will not show you its files. A bank might change its computer password every day (or every hour). See **log on, personal code.**

**PCB**: see **printed circuit board.**

**PD**: see **PENDOWN.**

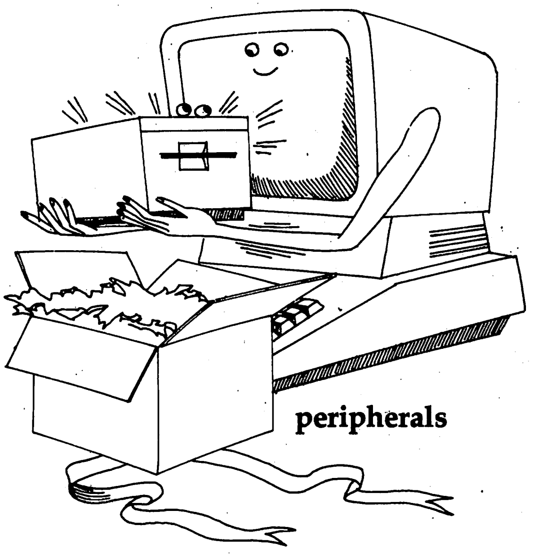
**PEEK** (pēk): in BASIC, the command that lets you look directly into your computer's memory. You type PEEK and the "address" number for one place in that memory. If that place has any data in it, your computer will show the data on its screen. (If you peek into memory space 1099, you might find a dollar sign ($) in it.) Your computer manual supplies the "addresses" for your computer's memory parts. See **POKE** for why you might want to PEEK!


**PEEK**

**PENDOWN** (pen'· doun'), **PD**: in LOGO, a command that tells a small figure (a "turtle") to draw a line as it moves on the computer screen. For examples of where the turtle might move, see **turtle, FORWARD,** and **LEFT.**

**PENUP** (pen' up'), **PU**: in LOGO, a command that tells a small figure (a "turtle") to move on the computer screen without drawing a line. (Most times, though, you want the turtle to draw.) See **turtle, PENDOWN,** and **BACK.**

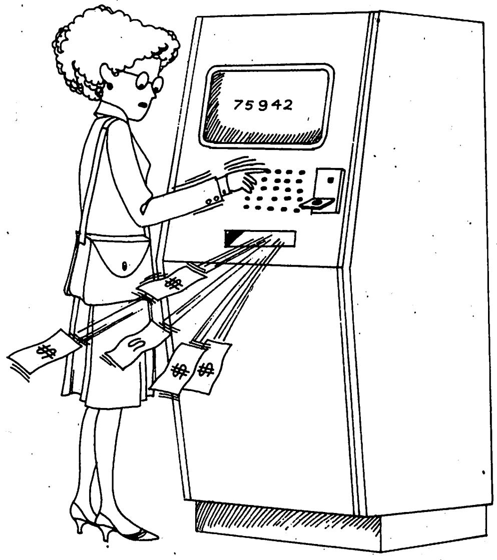
**peripherals** (pə rif' ər əls): extra equipment you buy for your computer. What's "extra" depends on the size and brand of computer. Almost all computers include a keyboard as a "necessary" item, and most include a TV-like screen. Large computers contain disk storage; but a disk drive is an "extra" for small personal computers. A printer and a joy stick are peripherals, but the "software" (instructions you give the computer) is not. For other peripherals, see **input, output.**


**peripherals**

**personal code** (pər' sən əl kōd'): an I.D. number you receive from your bank or another source. You might use it when you withdraw money from a cash machine. First, you put your bank card into the machine. Then you type your personal code into the computer. (Double protection for your money!) See **password.**

**personal computer** (pər' sən əl kəm pyū' tər): see **microcomputer.**

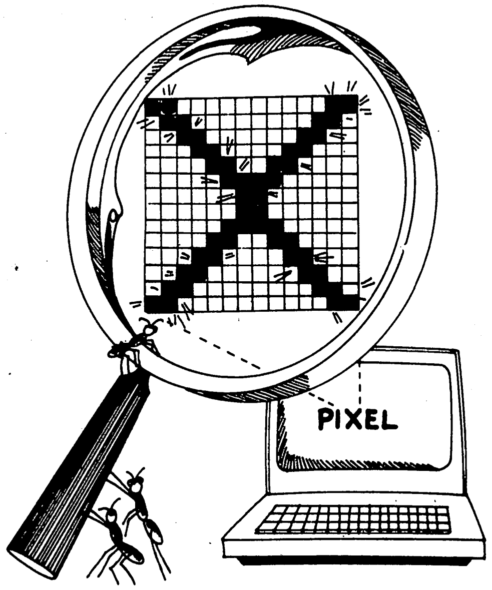
**pico-** (pē' kō), **p**: a prefix that changes a word to mean one trillionth part of something. See **picosecond.**



**personal code**

**picosecond** (pē′ kō sek ənd): one trillionth (.000000000001) of a second. (Even a computer struggles with this one.)

**PILOT** (pī′ lət): a "high-level language," used for education. It helps you to start writing programs with simple instructions. See **language, authoring language.**



pixel

**pixel** (pik′ səl): the smallest point of light on your computer screen. Your computer knows how many pixels to light up for a "T" or a "6." But if you want it to draw a picture on the screen, you have to tell your computer which pixels to use. You name a pixel by its column and row numbers (like 30 columns across, 8 rows down). Your computer manual will tell you the "coordinates" (pixel numbers) for your screen. See **animation, graphics.**

**PL/1, Programming Language 1**: a "high-level" computer language, used in science and in business. See **language.**

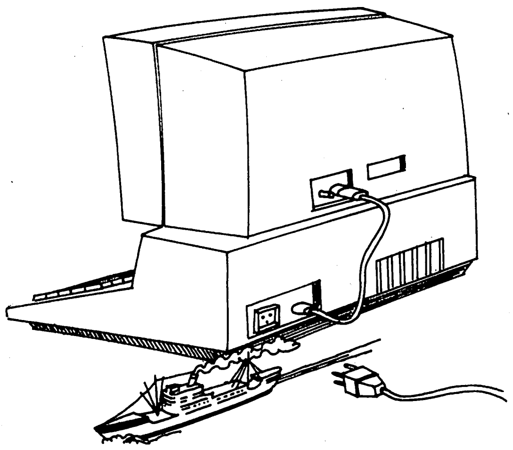
**PLOT** (plot): in BASIC, your way of telling the computer to light up a particular part of the screen. PLOT is good for drawing. To use this command, you have to know: (a) how many columns wide your screen is; (b) how many rows it has from top to bottom. Check your computer's manual to find out how it works on your computer.

**plotter** (plot′ ər): see **graphics plotter.**

**POKE** (pōk): in BASIC, the command that lets you choose where to put something in your computer's memory. If you want your computer to print a question mark in the center of your screen, in Apple BASIC, you would type:

```
POKE 1468, 191
```

The number 1468 is the center spot on the Apple screen. The number 191 is a code for Apple's question mark. Knowing how to POKE is important for writing games and "simulations." Your computer manual probably lists the "addresses" (poke numbers) for each spot on your computer screen. See **PEEK.**



port

**port** (pôrt): a socket, or an opening, in a computer where attachments can be plugged in. See **interface.**

**positive number** (poz′ ə tiv num′ bər): in mathematics, a number more than zero. A positive number is written with the plus sign (+) in front of it, or with no sign at all. The number 13 is a positive number. So is +13. See **negative number**.

**power** (poủ′ ər): **1.** the number of times you multiply a number by itself. When you see "$4^3$" it means "four, to the third power," or 4 times 4 times 4. (That's 64.) See **exponentiation**. **2.** the speed of a computer. Today, personal computers can carry out a half-million steps by the time you type the letter "a." **3.** electricity that runs a computer, as in "power on," or "power failure." See **electricity**.

**power**

**PRINT** (print): **1.** in word processing, your command to the computer to put a letter or some other document on paper. **2.** in BASIC and in LOGO, a command telling the computer to show something on the screen. In both languages, the computer's response depends on how you set up the PRINT command. Here is a BASIC command, followed by the computer's response:

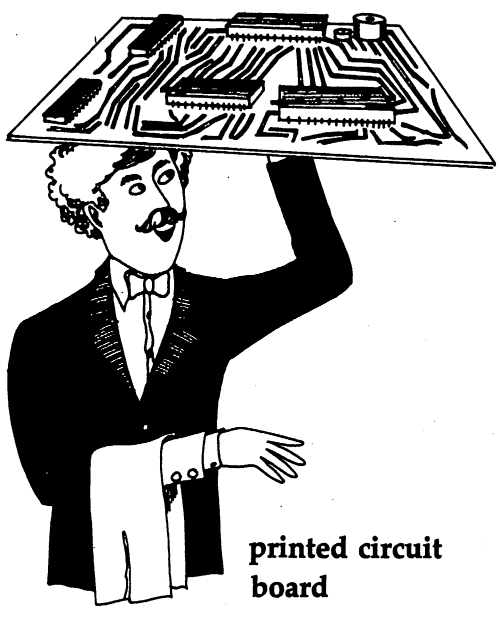
```
    PRINT 3*(4+5)−2
    25
```

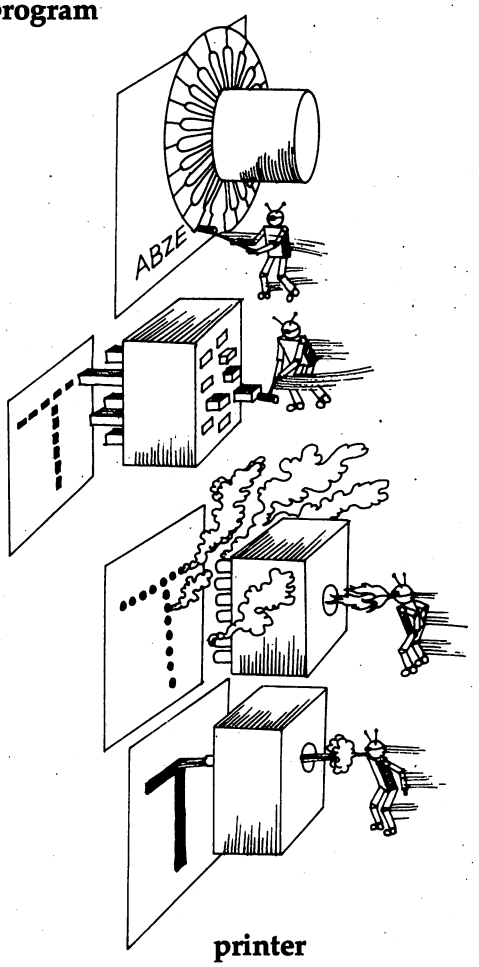
This PRINT command really says, "Do the work (add, multiply, and subtract), and show the result." But if you want the computer to print exactly what you say to it, you add a symbol to your PRINT command. In LOGO, you use brackets; in BASIC, quotation marks. Here's a sample LOGO command and the computer's response.

```
    PRINT [AM I A SMART COMPUTER?]
    AM I A SMART COMPUTER?
```

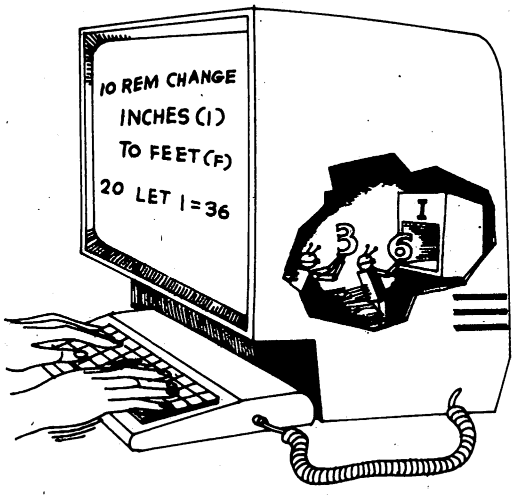
Most times. See **printer**.

**printed circuit board** (prin′ tid sər′ kit bôrd), **PCB**: in a computer, the thin board on which "chips" sit. The chips on a PCB usually include: (a) the microprocessor, which works on data; (b) a "ROM" chip with start-up instructions for the computer; (c) another chip or two ("RAMs") for storing your program while the microprocessor works on it. The PCB also has tracks ("printed circuits") for carrying electric pulses ("bits") from one chip to the other. See **chip, circuit, RAM, ROM**.


**printed circuit board**



printer



program

**printer** (prin′ tər): a device the computer uses to give you a paper copy of what it knows. There are many ways to classify printers: Some printers, for example, do one letter at a time. Others print a whole line at once; and others, a whole page. Some are "impact" printers which "hit" ribbon against the paper; others are "non-impact" printers. **1.** *Impact printers* include: (a) a **daisy wheel** printer, which has a circular device with each "character" (a, b, c, 1, 2, 3) attached at the end of a spoke. As the "wheel" turns, a small hammer hits each letter the computer wants, and the letter hits an ink ribbon against the paper. (b) a **dot-matrix** printer, which has no letters, just a cube (a "head") with rows of pins sticking out. To shape each letter, different sets of pins hit the ribbon against the paper. (Imagine holding a fistful of pencils over a piece of paper. If you push 7 of them down, they make 7 dots. If they're all in a column, they make the letter "I.") (c) some typewriters, if they can be connected to the computer. **2.** *Non-impact printers* include: (a) a **thermal matrix** printer, similar to dot-matrix. When hot pins come near the paper, they make small dark spots appear. These spots form letters. (b) an **ink-jet** printer, which sprays tiny jets of ink at the paper. The jets land in the form of perfect letters. (A very expensive kind of printer!)

**procedure** (prə sē′ jər): **1.** in general, steps that carry out a task; a "routine" for solving a problem. **2.** in writing programs, a rule for one particular kind of task; a small program. See **function, subroutine, program.**

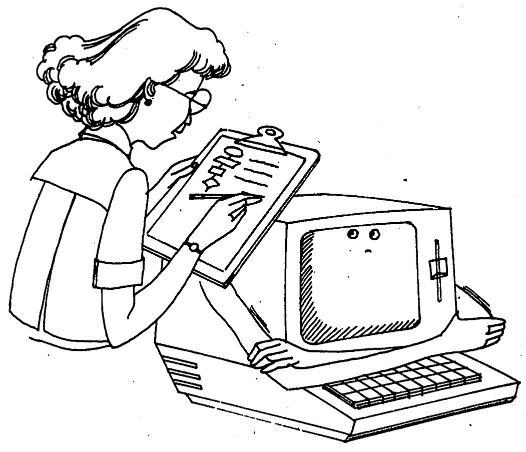
**processor** (pros′ es′ ər): see **central processing unit.**

**product code** (prod′ əkt kōd′): see **bar code.**

**program** (prō′ gram): computer "software"; instructions that tell a computer how to work. There are two major types of programs: *System* programs are "wired" into the computer. They tell it how to "wake up" and get going. *Application* programs are your instructions for one particular job. A spelling game would be an application program. So would a program to write music. You can get application programs from the manufacturer of your computer, on cassettes and tapes, from books and magazines, from friends, and from your own brain. How do you write a program? In most cases, you need to know a computer language such as BASIC. Your computer manual will give you clues, too. Here's a short program in BASIC:

```
10 REM CHANGE INCHES (I) TO FEET (F)
20 LET I = 36
30 LET F = I/12
40 PRINT F; "FEET"
50 END
```

Each line is numbered, usually by 10's. Each instruction, called a "statement," is on a separate line. On line 10, your REMark is "documentation," telling anyone who sees the instructions what they are for. On line 20, you give the computer a piece of "data" to work on; on line 30, you tell the computer how to work on it. Line 40 tells the computer how to report the answer to you. Most BASIC programs have an END statement like line 50. When you type RUN, the computer will follow your program's instructions. Instantly, you will see 3 FEET on the screen.



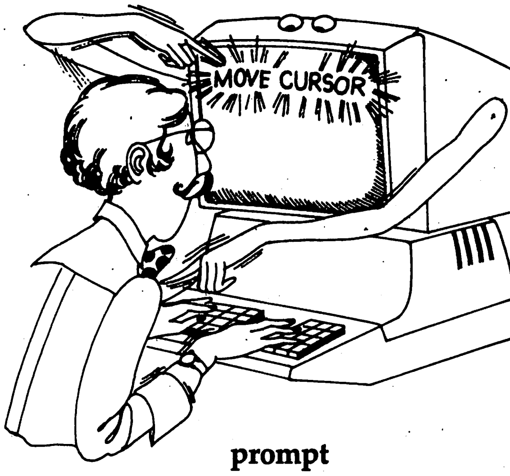
programmer

**programmable read-only memory** (prō gram' ə bəl rēd' ōn' lē mem' ər ē), **PROM:** see **read-only memory.**

**programmer** (prō' gram' ər): a person who writes instructions ("programs") for a computer. See **program.**

**PROM:** see **read-only memory.**

**prompt** (prompt): **1.** any message from a computer to the person using it. *The new program I wrote has a "prompt" message in it. It prints "Countdown" when my computer's memory space is running low.* **2.** a message from the computer that hints what your next step might be. In BASIC, your computer might show you a question mark (?) when it's hungry for another piece of data. The prompts in a computer "menu" help you select the place where you want to start working in a new program. See **menu.**



prompt

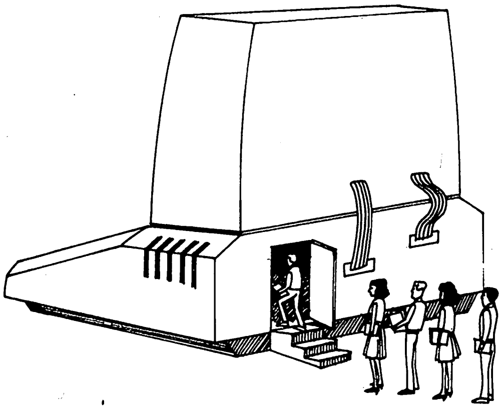
**PU:** see **PENUP.**

**pulse** (puls): see **electricity.**

**punched card** (puncht kärd): see **Hollerith code.**

## Q

**queue** (kyū): a lineup of work for one particular medium-size computer. A queue may include (a) programs waiting to be run (carried out); (b) incoming messages that need action; (c) outgoing messages waiting for an open line. The item waiting the longest, gets the next attention. See **first-in-first-out.**



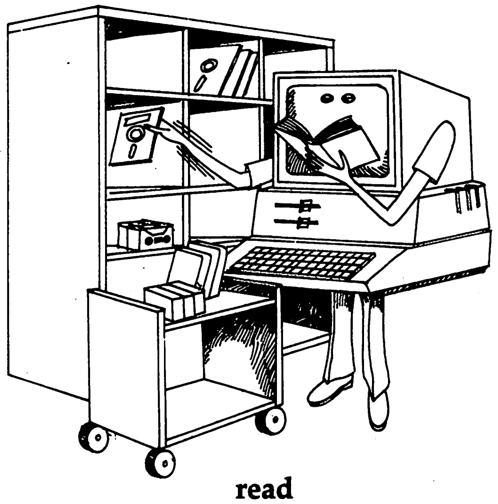
queue

**qwerty keyboard** (kwər' tē kē' bôrd): a computer keyboard with keys arranged the same way as in a standard typewriter. "Qwerty" comes from the first six letters in one of the rows. See **keyboard.**

## R

**RAM** (ram): see **random-access memory.**

**random-access memory** (ran' dəm ak' ses' mem' ər ē), **RAM:** the computer's "working" memory; its "temporary" memory. When you put instructions and data into a computer, they turn into tiny pulses of electric code which fill the RAM chips. While your computer is on, you can change anything in RAM; but RAM goes blank when you turn off the computer. If your program came from a disk or tape, you still have that copy. But if you typed it into the computer, you have only two choices: (a) Make a tape or disk copy before you turn off the computer; or (b) lose all your work. A computer's RAM has limits, so you have to check the "size" of a program before you buy one on a disk or tape. It takes about "1K" RAM to hold a program that is 40 lines long. (A "K," or kilobyte, is a piece of computer code.) See **memory, chip, disk, tape, cartridge, kilobyte, program, ROM.**



read

**read** (rēd): by a computer, to find data that was stored away, and get it ready to be used. Usually, a computer reads from a magnetic disk or tape, or from holes in "punched cards." When you tell a computer to read data, it finds the data and then changes the data into electric pulses. A computer reads in order to "write" data someplace else, maybe in the computer's "adder." See **write, tape, disk, Hollerith code, adder.**

**READ:** in LOGO, your command to the computer to take data from a magnetic disk and show it on the screen. READ makes the computer transfer a file of records to its active memory (not "LP" records, but recorded information). If you want to look at your file of science definitions, your READ command might look like this:

```
READ "SCIENCEWORD
```

(In LOGO, you need just one quotation mark). See **READPICT, workspace, file, SAVE.**

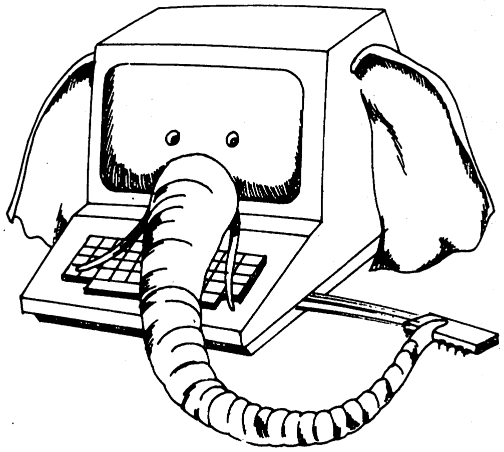
**READ . . . DATA** (rēd′ dā′ tə): in BASIC, a set of instructions telling the computer to use facts in the order they appear. The READ instruction is on one program line. The computer finds your DATA on another. For example:

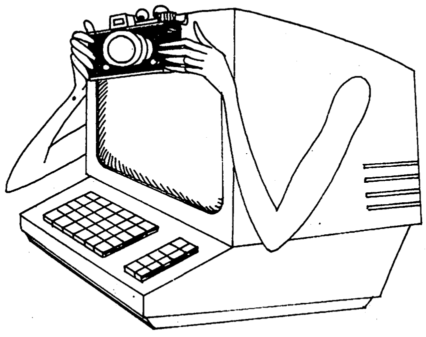
```
10 READ L, W
20 LET A = L*W
30 PRINT A
40 DATA 5, 10
50 END
```

This is a short program for finding the area of a rectangle. On the READ line, you tell the computer to find the number values of L and W. When it comes to the DATA line, your computer knows that "L" is "5" and "W" is "10". (DATA line items follow the same order as READ line items.) READ . . . DATA instructions are useful when you have long lists of numbers for the computer.

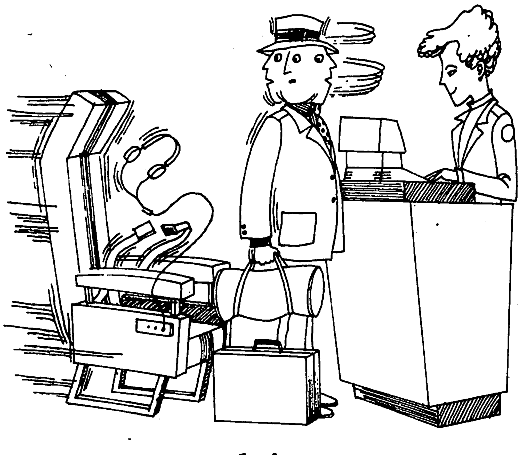
**read-only memory** (rēd′ ōn′ lē mem′ ə rē), **ROM:** the computer's "permanent" memory; the things a computer knows and doesn't forget. Where is this great memory? On a small silicon chip inside the computer. Circuits in a ROM chip always carry the same coded instructions. They tell the computer how to: (a) start itself up; (b) interpret a language like BASIC; (c) add and subtract; (d) check on the keyboard, screen, and other parts of the computer. You can buy extra ROMs for some computers. It is even possible to get an "EPROM" chip—a ROM chip that you can erase ("E") and reprogram ("P").

**READPICT** (rēd′ pikt′): in LOGO, your command to the computer to take a "picture" from a magnetic disk and show it on the screen. In LOGO, it's possible to draw pictures on the screen, and then save them on a disk. With READPICT, you get your drawing back from the disk. When the picture is on the screen again, you can add to it, change it, or just admire it. See **SAVEPICT, READ, turtle.**


**read-only memory**

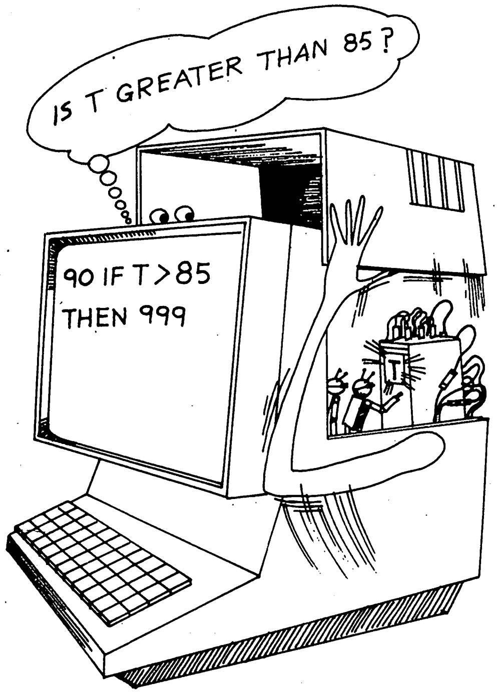

**READPICT**

**read/write head** (rēd rīt hed): the part of a computer attachment that "reads" data from a disk, and puts it there, too. The head is part of a disk drive. Its job is similar to the job of a needle on a stereo. As the disk turns, the read/write head moves over it. The head picks up magnetic signals from the disk surface and converts them into tiny electric pulses ("bits")—the only code a computer can work with. The read/write head also goes in reverse, putting magnetic signals on the disk to store information. The head can find any piece of data on a disk in a split second.


real-time

**real-time** (rēl tīm): able to give you results almost as soon as you supply information. Airlines use real-time computer systems. Walk into any airline reservation office and ask about an available plane seat. Within about three seconds, the computer terminal will tell you if you can fly when you would like. That terminal, and others around the country, may all be linked to one main computer in Chicago. The main computer may receive questions from all the terminals at the same time. It checks passenger lists and schedules, and answers all the questions within the same three seconds. See **batch processing** for a different way of working.

**register** (rej' i stər): a place where a computer stores data; a storage place in the computer's active memory. There are different kinds of registers, such as (a) an "instruction" register, to hold instructions from your program; (b) a "shift" register, to move electric pulses for multiplication; and (c) an "accumulator," where numbers are added. See **memory, shift, accumulator.**



relational operator

**relational operator** (ri lā' shən əl op' ə rā' tər): a sign that tells the computer to compare two things. There are six relational operators in BASIC:

| *Relational Operator* | *Its Meaning* |
|---|---|
| F = 109 | F is equal to 109 |
| F < 109 | F is less than 109 |
| F > 109 | F is greater than 109 |
| F <= 109 | F is less than or equal to 109 |
| F >= 109 | F is greater than or equal to 109 |
| F <> 109 | F is not equal to 109 |

The first relational operator is used in many BASIC instructions. For example:

```
30 LET R = 2*(A + N−1)
```

This statement tells the computer to do the work after the "equal" sign, and then store the answer in "R." Relational operators are also used for setting up "tests." For example:

```
90 IF T < 85 THEN 999
```

You tell the computer to test "T," a place in its memory. (If T holds a number larger than 85, the computer will go to line 999.) See **operator, branch.**

**REM, REMark:** see **program.**

**REPEAT** (ri pēt′): in LOGO, a command to list words or drawings as many times as you want. For example, if you want the computer to print "America" twice, you would type:

```
REPEAT 2 [PRINT "AMERICA]
```

You can invent very beautiful designs using REPEAT and LOGO's "turtle" drawings. See **turtle, LOGO.**

WRITE - I WILL BE QUIET 500 TIMES
REPEAT 500 [PRINT "I WILL BE QUIET]
REPEAT

**RESET** (rē set′): a very powerful key, on a personal computer. When you press it, everything stops, "empties out," goes blank. Some manufacturers tell you to try RESET, if you don't know what else to do. Others say, "Tape the key over! Never use it." Check your manual on this one.

**RETURN** (ri tərn′): **1.** a key you touch to put a new piece of information or data into the computer. You type the information first, then press the RETURN key. **2.** in BASIC, an instruction that tells the computer to pick up where it left off in the main program. It's something like a car that takes a detour, and then "returns" to the main road. See **GOSUB, subroutine.**

RESET

**RIGHT** (rīt), **RT**: in LOGO, a command that rotates a small figure (a "turtle") on the computer screen. Your command to turn RIGHT is followed by a number. RIGHT 180 would make the turtle turn 180 degrees to the right. (That's an "about-face" in the army.) You would need another command to make turtle march forward. See **turtle, LOGO, LEFT, FORWARD.**

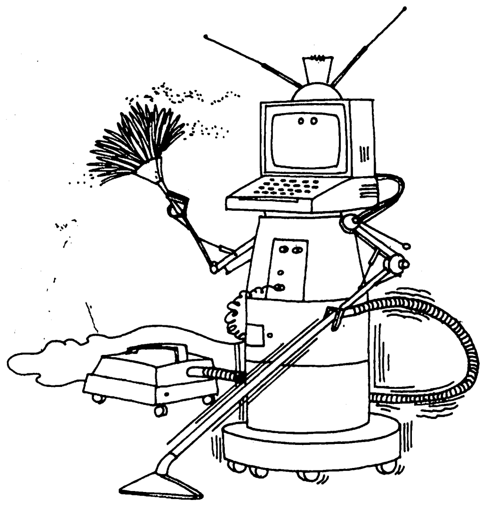
**RIGHT$** (rīt string—the $ symbol stands for "string"): in BASIC, a command to the computer to print part of a "string" of letters in its memory. The part that you want is on the right. Suppose you tell your computer the date of a holiday. You put it between quotation marks and give it a "string variable" name (H$).

```
LET H$ ="NEW YEAR: 010186"
```

Here's how you ask the computer to print just the date. (The second line is what your computer would print.)

```
PRINT RIGHT$(H$, 6)
010186
```

"RIGHT$" tells computer to read only the right part of H$. And the "6" tells the computer to print the last six "characters" (the numbers). A business program might use RIGHT$ to pull zip codes from lists of addresses. See **LEFT$** and **MID$**. See **string variable, list.**

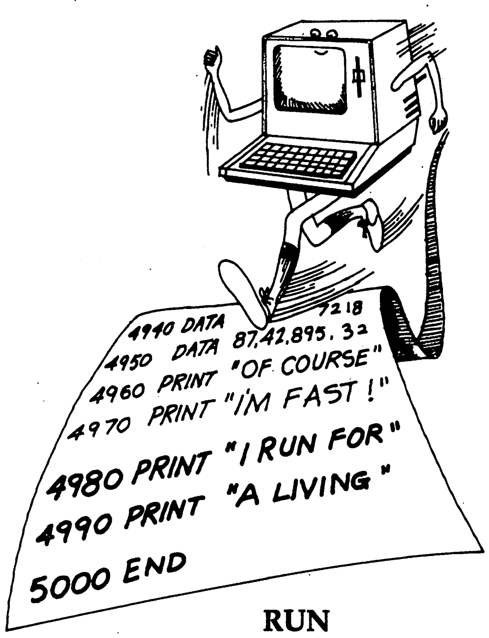

robot

**robot** (rō' bət): a device that does things we call human. Robots contain computers that tell them how to "touch," "see," and move objects near them. As they move, robots gather data about where they are. Using this data, their computers give robots new instructions. If you bumped into a chair, your brain would probably send out an "Ouch!" A robot would skip the ouch and just remember not to go near the chair again. Robots work in factories today. Tomorrow, our homes. See **artificial intelligence, bionics.**

**ROM:** see **read-only memory**

**RS-232:** a popular "interface" or "connector" that allows a computer to work with a printer and other attachments. See **interface.**

**RT:** see **RIGHT.**



RUN

**RUN** (run): **1.** in BASIC, your command to the computer to carry out ("execute") a set of instructions. Your program may take months to write and a couple of days to type into the computer. Finally, you type RUN, and your computer completes the program in two seconds. That's really "running." See **program.** **2.** in LOGO, your command to the computer to print a list of letters and numbers ("characters"). Here's a short example.

```
    RUN [PRINT [YOU'RE A GOOD FRIEND]]
    YOU'RE A GOOD FRIEND
```

The second line shows what your computer would print.

## S

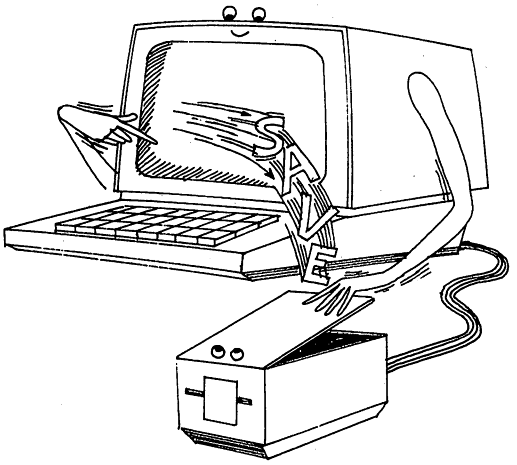
**SAVE** (sāv): a command to the computer to transfer your work from the screen to a disk or tape. You don't have to save a program that you buy—it is already stored on a disk, tape, or cartridge. But if you type a program you created into the keyboard and you don't want to lose it, then you must SAVE it before turning off the computer. Here's how you save a spelling program in BASIC:

```
    SAVE GENIUSINSPELLING
```

And here's how, in LOGO:

```
    SAVE "GENIUSINSPELLING
```

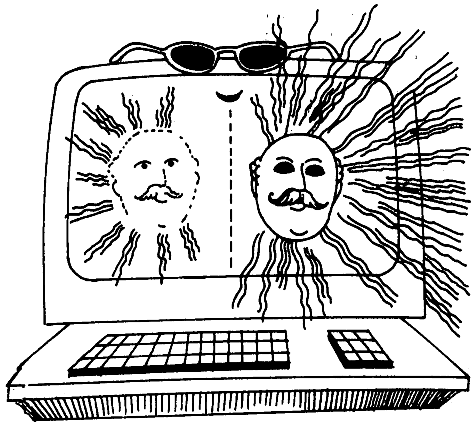
(That's right. Only one quotation mark in LOGO.) See **LOAD, disk, tape, SAVEPICT**, and **READ**.



SAVE

**SAVEPICT** (sāv' pikt): in LOGO, your command to the computer to remove a picture from the screen and put it on a magnetic disk. In LOGO, you can draw pictures on the screen with the help of a small figure called "turtle." See **SAVE, turtle, READPICT**.

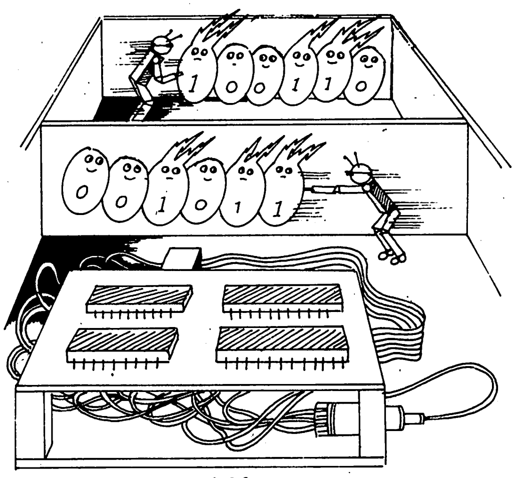
**screen resolution** (skrēn' rez' ə lū' shən): the amount of detail your computer can get in a drawing on its screen. A screen that shows many small details has "high resolution." A screen that shows only chunky drawings has "low resolution." Resolution is a problem for your computer's memory, not its screen. It takes only one instruction to light the whole screen. It takes four instructions to light each corner. If your computer has to light 100 "points" of light ("pixels"), it needs memory space for 100 instructions, and so on. See **pixel**.



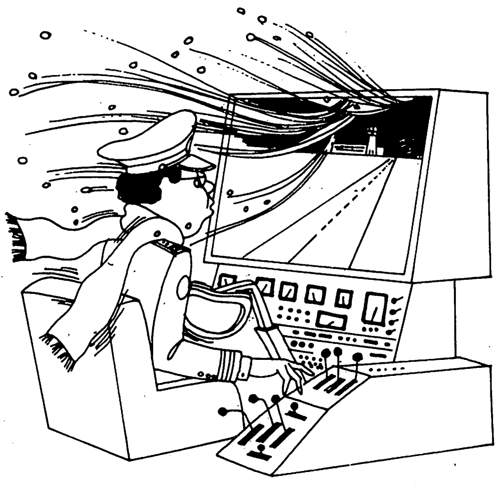
screen resolution

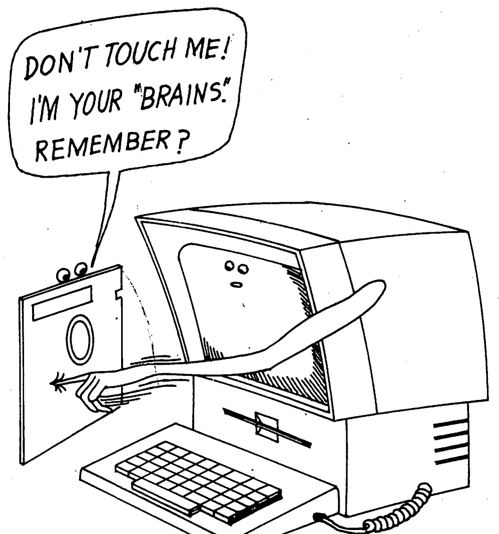
**semiconductor** (sem' ē kən duk' tər): any solid material in which the flow of electricity can easily be controlled. Silicon is a semiconductor material used in computer "chips." See **transistor, circuit, chip, silicon**.

**shared time** (shārd tīm): see **time sharing**.



shift


simulation



software

**shift** (shift): **1.** to change from typing small ("lower-case") letters to capital ("upper-case") letters. You can shift from typing this way to TYPING THIS WAY. Or This Way. (Your computer manual may tell you other ways to use your shift key.) **2.** in word processing, to move a word to the right, when you insert another word before it. **3.** in a computer, to move a set of "bits" (electric pulses) to the right or left. Computers shift bits when they multiply and divide. See **binary system, register.**

**silicon** (sil′ i kən), **Si:** a natural element that occurs in sand and rock. It is the main ingredient in a "chip," or semiconductor. A chip is the work area in a computer. See **chip, Silicon Valley.**

**silicon chip** (sil′ i kən chip′): see **chip.**

**Silicon Valley** (sil′ i kən val′ ē): a term sometimes used for an area near San Francisco, California, which is a leading center of computer research and manufacture. Silicon is used in making "chips." See **silicon, chip.**

**simulation** (sim′ yə lā′ shən): a model of how things work under different conditions. Simulations help us to learn about things that are hard to bring into a classroom. Computer simulations teach people how to fly planes: The student can "see" an "airstrip" on the screen, and prepare to "land" a plane in a "snowstorm." Simulations are also used in research. For example, research into the mistakes that students make in simulated landings might show the need for safer landing equipment. Video games are like simulations, but their purpose is not to teach.

**smart computer** (smärt′ kəm pyū′ tər): see **intelligent computer.**

**soft copy** (sôft′ kop′ ē): what we see on a computer screen. We call this "soft" copy because it disappears when the computer is turned off. If a computer is attached to a printer, it gives us "hard" copy on paper. Hard copy and soft copy are forms of computer "output." See **hard copy, output.**

**software** (sôft′ wār): **1.** all the instructions and data in a computer; the programs that a computer carries out. Software includes programs you write and programs you buy on disks and tapes. (The computer you run it on is "hardware.") See **program, data.** **2.** all the electric pulses ("bits") in a computer. Everything a computer works on has to be in electric pulses which form a code. See **bit, byte, word.**

**space bar** (spās bär): **1.** in most computers, the key you use to put spaces between words.

Thespacebarwasnotusedinthissentence.

**2.** in some computers, a key for erasing letters and words you have already typed. For example, to get rid of the sample sentence in definition 1, you could: (a) back up to the beginning of the sentence; (b) press the space bar; (c) continue pressing it until the whole sentence disappears. For another way to get rid of a sentence, see **DELETE.**

**speech synthesis** (spēch′ sin′ thə sis): the way some computers put sounds together in order to "talk" to you. Computers learn "sounds," not words. But they combine sounds to make words, depending on the instructions they get. Suppose you use a phone booth to call home from the next town. If you talk more than a minute or two, you'll hear something like this from the phone computer: "De-pos-it- for-ty-cents- for-the-next- five-min-utes." It's speech synthesis. See **voice recognition.**

**speech synthesis**

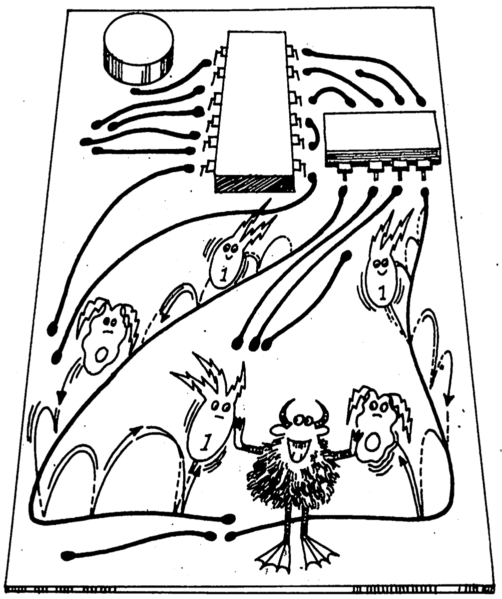
**SQR, SQuare Root:** a code in BASIC that tells the computer to find the square root of a number. SQR(25) will produce 5. SQR(100) will get you 10. See **library function.**

**SQRT, SQuare RooT:** in LOGO, a code that tells the computer to find the square root of a number. SQRT 25 is 5, to a computer that reads LOGO.

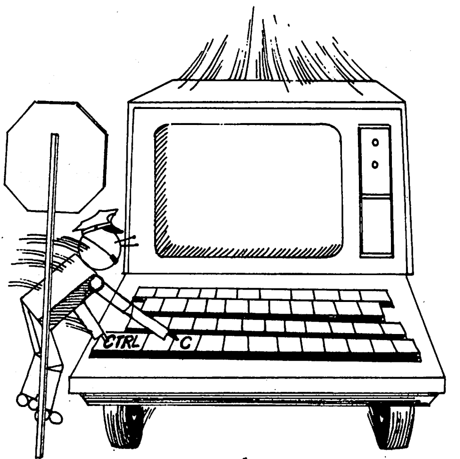
**statement** (stāt′ mənt): an instruction; a brief description of one task you want the computer to do in a program. This program might include 4 statements. Or 40. Or 400. See **instruction.**

**statement number** (stāt′ mənt num′ bər): see **line number.**

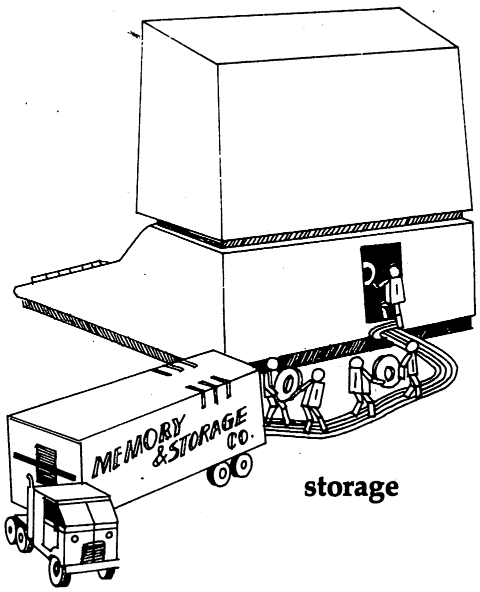
**static electricity** (stat′ ik i lek′ tris′ ə tē): electricity that does not flow as it should; electricity that is interrupted in its path. Static can occur when you rewind a computer tape very fast in dry weather. This sometimes destroys the magnetic code on a tape. (It's enough to turn a chess game into tic-tac-toe.)



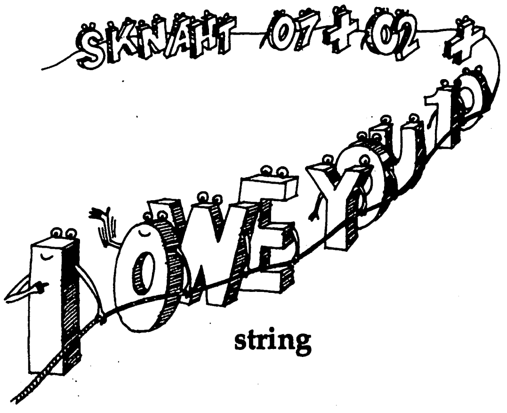
**static electricity**



stop



storage



string

**step** (step): **1.** an instruction you give the computer; a statement. See **statement, instruction.** **2.** an instruction the computer gives to itself. Suppose you tell the computer to LET B = 6. That's one "step" from you. But the computer has to "decode" "L" (one step), decode "E" (another step), decode "T" (another step), and so on. Maybe 10 or 15 "steps" altogether for the computer.

**STEP:** see **increment.**

**stop** (stop): **1.** in BASIC, to halt the computer's work. Why? Maybe you want to think about how a program is turning out. How? Hold the control key (CTRL) while you touch "C." (Different brands of computers have different ways of stopping.) See **control key.** **2.** in LOGO, to switch the computer's attention from one set of steps to another.

**storage** (stôr' ij): places where the computer keeps its information. A computer's "main" storage is on tiny chips of silicon. These chips are inside the keyboard in a personal computer. "Backing" storage (tapes and disks) are outside the keyboard. When you want to use a disk program, you "feed" it into main storage, first. In main storage, computer data is a stream of electric pulses. In backing storage, data is usually in magnetic code. The term "storage" is often used instead of "memory" (which is like saying, "use your head," when we mean, "use your brain."

**store** (stōr): see **storage.**

**string** (string): **1.** a series of items, one following after the other (for example, a string of electric pulses, or "bits"). See **bit.** **2.** a series of "characters" (letters, numbers, etc.) that the computer stores as one fact in its memory. Here's a string from a BASIC program. (The string is everything between the quotation marks.)

```
30 PRINT "I OWE YOU 10 + 20 + 70 THANKS!"
```

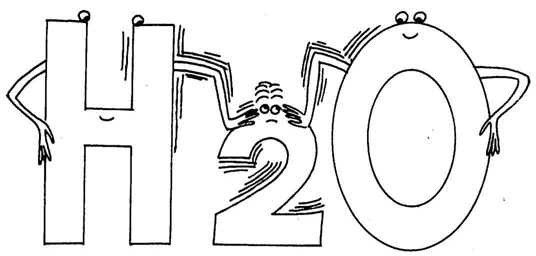
When it runs your program, the computer will print this string as you typed it. Your computer will not add numbers inside a string, even though you use a "work" symbol, the plus sign (+). See **string variable, word** (definition 2).

**string variable** (string' vär' ē ə bəl): a place in its memory where the computer stores a series of items (a string) you don't want it to change. A "variable" is the name for a memory space. (In this command, LET R = 19, the variable is R.) To write a string variable, you add the dollar sign ($) to a variable. Some examples: R$, C$, M$. Suppose you want your computer to store the title of a play you're writing. Here's how to enter the title as a string variable. (The last line is what the computer will print.)

```
    LET M$ = "OUR 3 + 2 FAMILY"
    PRINT M$
    OUR 3 + 2 FAMILY
```

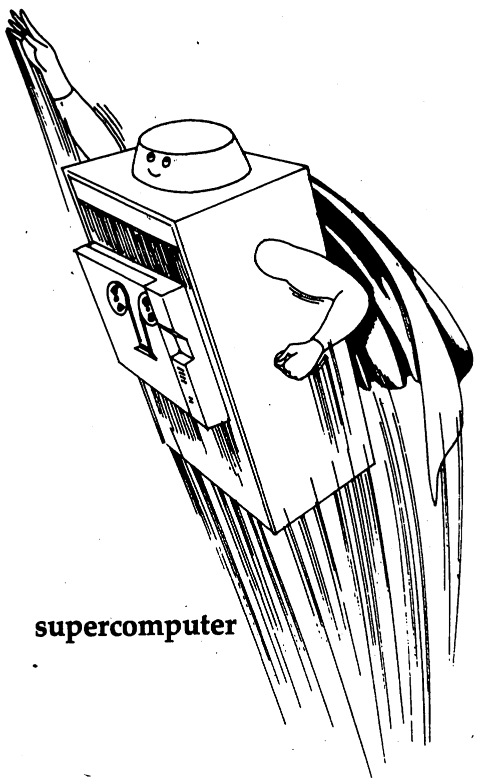
Though you use a plus sign, your computer will not change anything inside a string variable. See **variable, string**.

**subroutine** (sub′ rū̅ tē̅n′): a set of instructions that the computer can switch to, in the middle of a program. Subroutines save time for a program writer. Suppose a coach wants a computer to rate and rank team members once a month. The program for this might include a 13-step formula to be used over and over. (Maybe it adds player points, exercise hours, etc.) The coach can write this formula once, and include it in the program for producing a monthly report. Each time the computer comes to a player's name, it will go back to the formula (a "subroutine"). How do you tell a computer to go back to a subroutine? In BASIC, you say "GOSUB," of course. See **GOSUB**.

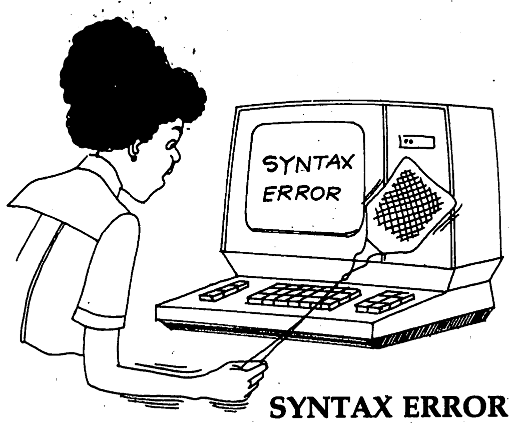

subscript

**subscript** (sub′ skript): **1.** generally, a number that you write to the right of another number or letter, and a little below it. It is used in math and science. For example, the symbol for water ($H_2O$) includes the subscript, "2." **2.** the number for one item in a computer list. If your computer holds a list (C) of seven checks you wrote in May, the seventh check would be C(7). See **list**.

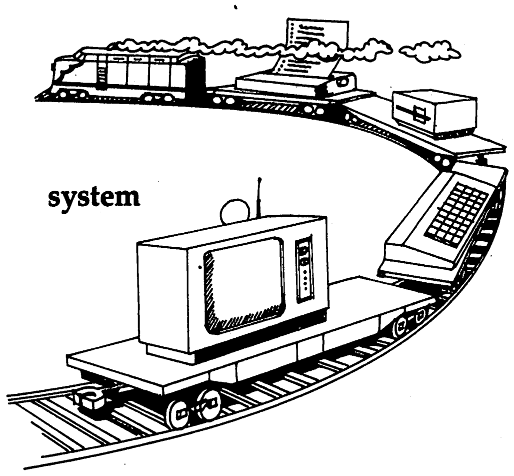
**supercomputer** (sū̅′ pər kəm pyū̅′ tər): a "mainframe" computer that handles more instructions per second than other computers. How many instructions? That changes every year or so. Today's supercomputer might handle more than 150 million instructions a second. Tomorrow's model might handle 200 million, and so on. Companies in the United States and Japan compete to produce the next supercomputer. See **computer**.


supercomputer

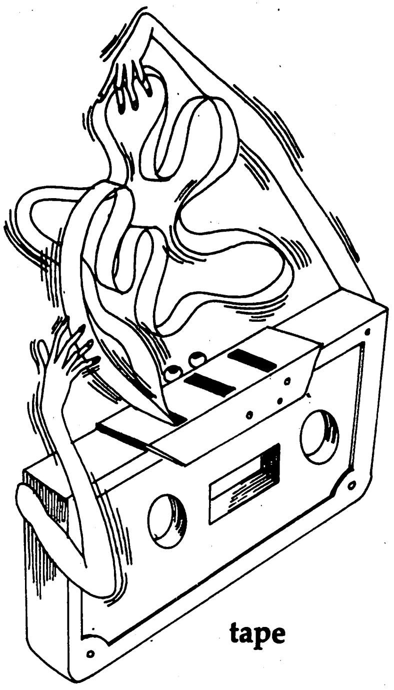
**symbols** (sim′ bəls): see **arithmetic operator, logic operator, relational operator**.



SYNTAX ERROR



system



tape

**syntax** (sin′ taks): **1.** in human language, the rules that tell us how to put words together in a sentence. For example, we say, "She likes me," not "She likes I." **2.** with computers, the rules that tell us how to use words and symbols in a program statement. See **SYNTAX ERROR, bug.**

**SYNTAX ERROR** (sin′ taks er′ ər): in BASIC, your computer's way of saying it doesn't understand you. When you see SYNTAX ERROR on your screen, it's a good idea to check for "bugs," or program errors. Bugs include: (a) misspelled words in your program, (b) missing command words, (c) incomplete statements. See **syntax, bug, prompt.**

**synthesizer** (sin′ thə sī′ zer): a device that takes instructions from a computer and produces sounds. Synthesizers allow some computers to "talk," play music, and produce sounds like a ringing bell. See **speech synthesis.**

**system** (sis′ təm): units, parts, or things that relate to one another as they work. Earth is part of a solar system. There are railroad systems and stereo systems. And there are computer systems. Most home computer systems have a keyboard and a TV-like screen. Others also include a disk drive and a printer. A part from one system may not work with another system.

## T

T: see **tera-**.

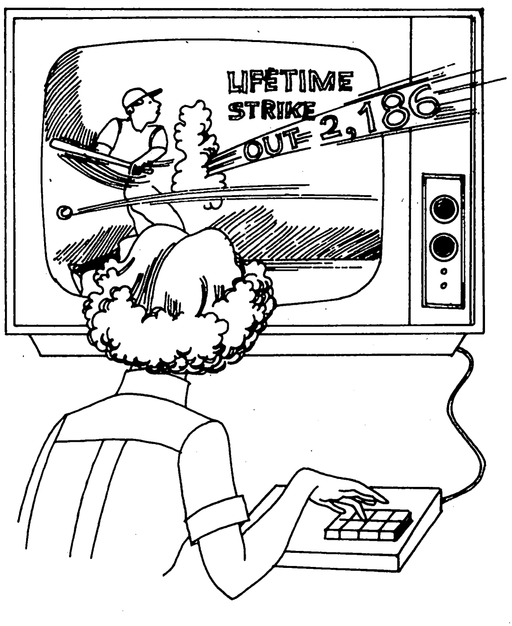
**TAB** (tab), **TAB**ulator key: a key you touch to let the computer know where to type the next number or letter. The place is a spot you picked out earlier, and placed in your computer's memory. TAB is often used to start a new paragraph some extra spaces from the margin.

**table** (tā′ bəl): an organization of facts; a way of arranging data so that you can easily find a single item. One example, a table of contents in a book. See **list** and **array** for different kinds of tables.

**tape** (tāp): usually, a thin "ribbon" of plastic on which you can store computer programs in code form. It is the least expensive form of backup memory a computer can use. It is also the slowest. You can buy blank tapes to record your own programs, and you can buy tape programs, "ready-made." See **disk, storage.**

**telecommunication** (tel′ ə kə myū′ ni kā′ shən): a computer message that travels on a radio, satellite, telegraph, or telephone signal. Usually, it goes by telephone. When one computer wants to talk to another, it needs a "modem." That's a device that changes computer signals into telephone signals, and vice versa. See **modem.**

**teletext** (tel′ ə tekst): a system for sending "print" information as part of a TV broadcast. Teletext allows you to switch from a TV picture to something you want to read on the screen. For example: If you were watching a TV baseball game, you could press a number key and look at the pitcher's rating this season. If you want teletext, you'll probably have to buy or rent a "decoder" to pick up the teletext signal. See **videotext.**



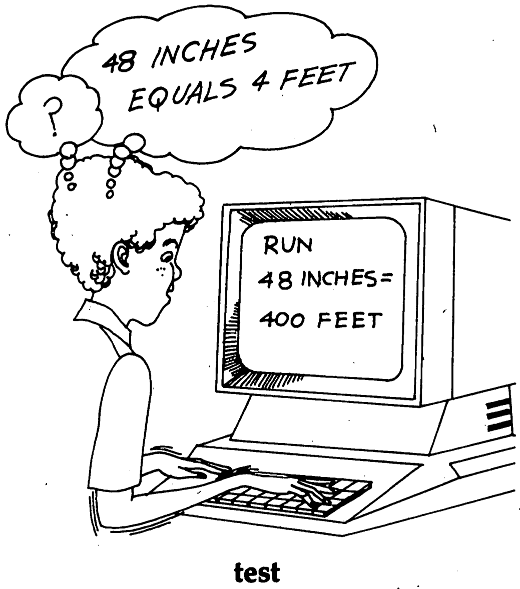
**teletext**

**tera-** (ter′ ə), **T**: a prefix that changes a word to mean a trillion times more. There aren't many examples of words built with tera-. Probably, because humans haven't thought about trillions very much . . . until computers came along. See **terabit.**

**terabit** (ter′ ə bit): a trillion electric pulses ("bits"). What's a trillion? One example: If you spent $100 per second (PER SECOND) for the rest of your life, you'd hardly make a dent in a trillion dollars. It would take more than 300 years to spend it at that rate. See **bit.**

**terminal** (tûr′ mə nəl): the equipment you use to give data to a computer, and get results from it. Usually, a terminal has a keyboard and a screen, but some terminals do not have keyboards. The screen lists five or six choices, you touch the choice you like, and the computer shows you a new list. Someday, all you'll have to do is talk to your terminal.

**test** (test): a trial run for a new program, to see if there are any mistakes in it. To be sure it's a test, you have to know the answer the computer should give you. Suppose you "invented" a program to convert inches to feet. Before you test the program with a sample problem, you should know the correct answer. If the computer does not get that answer, you would have to rework your program instructions, and test again.



**test**

**TEST:** in LOGO, a command that tells the computer to find out if a statement is true or false. Using the command in the first line sets up the test. (For example, you might want the computer to check the number it has in its memory for "K." Is that number greater than 0?) The next two lines tell the computer what to do if the answer is "yes" (TRUE), and if the answer is "no" (FALSE):

```
TEST: K > 0
IFTRUE OUTPUT "POSITIVE
IFFALSE OUTPUT "NEGATIVE
```

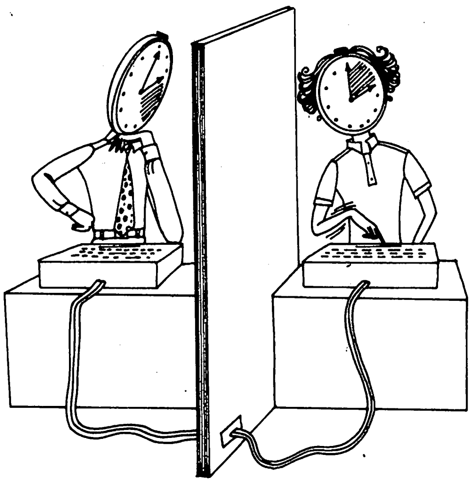
Suppose K is 45. What will the computer print? See **branch**.

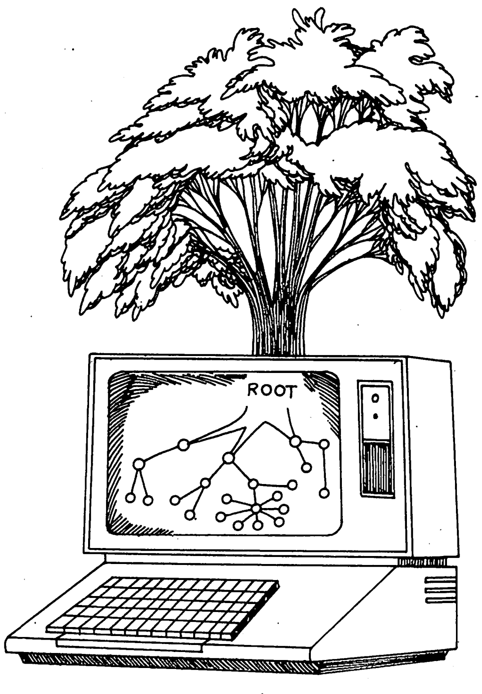
**THEN** (then): see **IF . . . THEN**.

**time sharing** (tīm' shār' ing): **1.** a plan that allows several people to use one large computer at the same time. Each user has a separate terminal (screen plus keyboard). These terminals may be in different rooms or different cities. All the terminals are in contact with the central computer. **2.** the way a central computer deals with several terminals at the same time. Suppose you and several friends are at different terminals in your school. All of you can work on the same math program at the same time. The school's central computer feeds a split second of the program to you, a split second to one friend, a split second to the next, and so on. It happens so fast that you don't notice any interruption. Time sharing is really "time slicing"!

**transistor** (tran zis' tər): a part of a computer "chip" that controls the flow of electric pulses ("bits"). (A chip is where your computer instructions are carried out.) Inside a computer, your instructions are coded as bits, and bits race at almost the speed of light. Transistors organize all this racing. They act like police who stand in the middle of traffic. They keep one set of bits moving, stop another, combine two more, etc. There are thousands of transistors on a tiny, quarter-inch chip. Together they make all the answers come out right. See **semiconductor, chip, circuit, logic gate**.

**tree** (trē): a diagram of how information is sorted and stored in a computer. A computer tree shows how a large group of information is divided into smaller pieces of information the way a root extends from a tree in nature. Subdivisions coming from the main root are called "members." But

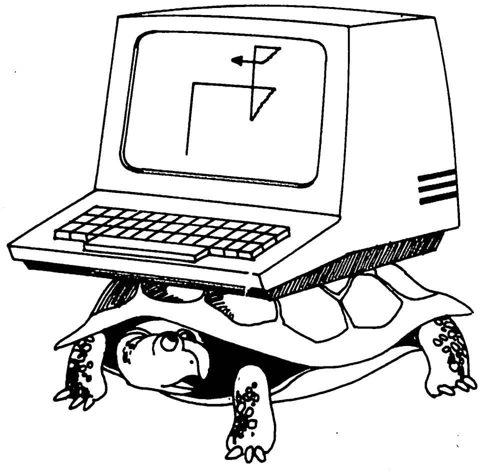

**time sharing**



**tree**

members also become "owners" if they have offshoots, too. Every member has its own code name, which includes part of its owner's name. That's why a computer can track down any single piece of information on a "tree." A computer tree might be used for the records of everyone in your school. The name of the school would be the main root, leading to subroots for grades, classes, and individual students. To get to your record, the computer would use the code name for your school, grade, class, and you. It's like getting a postcard in the mail. The zip code gets it to your town; the address gets it to your house (or apartment); and your name gets it to you. Maybe someday, we'll have code names, too.

**turtle** (tûrt' l): in LOGO, a small figure that draws on your computer screen. You type different commands to get the turtle moving. Turtle is a powerful teacher. Just a few commands get results. Almost without noticing, you learn to write programs. You learn some geometry, too, after building different shapes with turtle's help. See **FORWARD** and **LEFT**, for examples of turtle commands.


**turtle**

## U

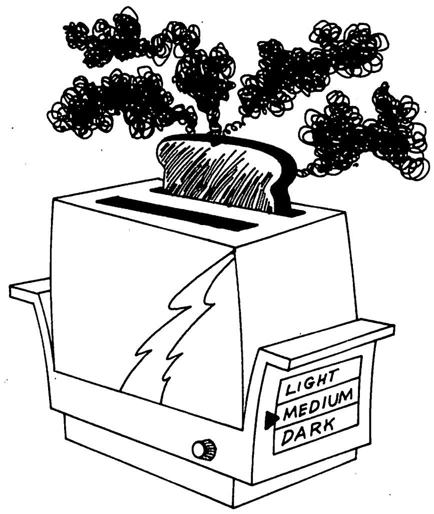
**ULC**: see **upper and lower case.**

**unconditional branch** (un' kən dish' ən əl branch'): see **branch** (definition 2).

**unconditional jump** (un' kən dish' ən əl jump'): see **branch** (definition 2).

**unconditional transfer** (un' kən dish' ən əl trans' fər): see **branch.**

**unintelligent computer** (un' in tel' ə jənt kəm pyū' tər): not able to take new instructions. Computers that operate some appliances are unintelligent. All they can do is what they learned at the factory. Take, for example, a toaster. It may have three or four settings (dark, medium, light, just-warm-it). You can't tell it to make French toast, or to select rye bread on Tuesday. See **intelligent computer.**



**unintelligent**

**universal product code** (yū' nə vur' səl prod' əkt kōd'), **UPC**: see **bar code.**

**unload** (un lōd′): **1.** to remove work from the computer. **2.** to remove the "head" of the disk drive from a magnetic disk. (The head "reads" information from the disk.) See **load, disk, read/write head.**

**UPC,** Universal Product Code: see **bar code.**

**upper and lower case** (up′ ər and lō′ ər kās′), **ULC:** the capitals ("CAPS") and "small" letters used in writing and printing words. THIS IS UPPER CASE; this is lower case.

**up time** (up tīm): the time when a computer system can be used. (No repairs needed!) See **down time.**

user-defined function

**user-defined function** (yü′ zer di fīnd′ fungk′ shən): any rule you teach the computer, for solving a problem. In BASIC, the code name for this kind of rule includes FN, plus a third letter that you choose. Suppose you invent a rule to figure out sales tax (T). The rule might be to multiply any sales amount (S) by .05. You could teach the computer your rule on line 40 of your program:

```
40 DEF FNT(S) = S*.05
```

You're saying to your computer, "Here's the rule (DEF FN) for tax (T) on sales (S)." Later, you might use the rule this way:

```
100 LET S = 10.55
110 LET A = FNT(S)
```

When you "run" your program, and the computer comes to line 110, it will find the sales tax, using the rule you taught it in line 40. See **DEF FN—, function.**

user-friendly computer

**user-friendly computer** (yü′ zer frend′ lē kəm pyü′tər): not a computer that wants to be your pal, but one that is easy to use. (At least, that's what the ad says.) See **metaphor.**

## V

**variable** (vār′ ē ə bəl): one space in your computer's memory; a place where one piece of data can be stored. (The piece of data stored in a space can "vary.") The name of a variable is usually one letter of the alphabet. In BASIC, here's how you would store "33" in variable J:

```
30 LET J = 33
```

On some computers, you can use a combination "letter-number" (such as J8) for the name of your variable. You can have up to nine such combinations (J1, J2, J3 . . .

J9). With 26 letters in the alphabet, that gives you a lot of memory slots to fill. What happens if you want to store a "list" of items in your computer's memory? It depends on the kind of list. See **list** for one possibility. And see **string variable** for another. See also **array**.

**VDU**: see **visual display unit**.

**video** (vid′ ē ō′): **1.** the "picture" part of a television broadcast. ("Sound" is the other part.) **2.** equipment that produces pictures. A cathode-ray tube in a computer is a video device. See **visual display unit**.

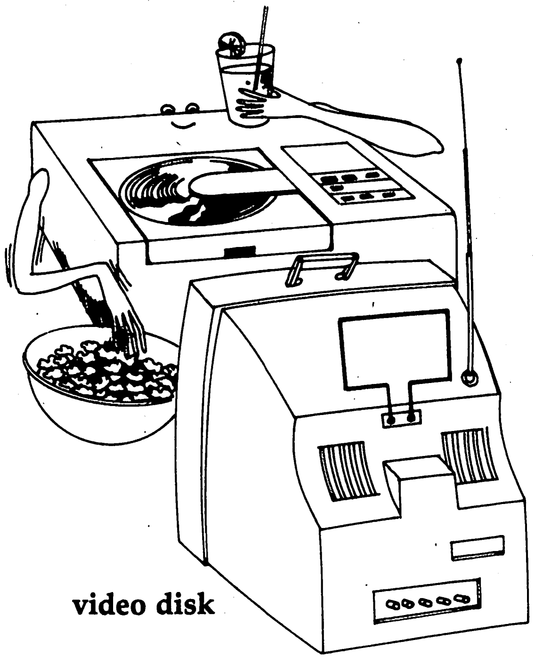
**video disk** (vid′ ē ō′ disk′): a round, plastic object that resembles a stereo record. Most video disks are now used for showing movies and other entertainment at home. But they could be used for education purposes, too. On a video disk the size of an LP, you could store about 750,000 pages like this one, pictures included. One disk would be a small library! Someday a video disk may hold up to 1,000 computer programs.

**videotext** (vid′ ē ō tekst′): see **viewdata**.

**viewdata** (vyū′ dā′ tə): a system for giving and getting information by using your TV, telephone, and a small computer keypad. Do you want a new pair of skis? Just dial a "viewdata" number. Then look at the sales catalog that appears on your TV screen. Next, press a few numbers on a keypad, and your skis are on their way to your home. Viewdata can be used in other ways, too, such as for taking courses or voting on local issues. Viewdata is just being tested in the United States. But soon, you may be dialing pizza from your TV . . . with extra cheese.

**visual display unit** (vizh′ ū əl dis plā′ yū′ nit), **VDU**: a TV-like screen. The VDU lets you see what you and the computer are "saying" to each other. See **cathode-ray tube**.

**voice recognition** (vois′ rek′ əg nish′ ən): the ability of some computers to "hear" you when you speak. It is very difficult to set up computer "codes" for all the different sounds in a person's speech. But a computer can be programmed to recognize a particular voice and a few words (like "Open the door!"). "Listening computers" would be great security guards. Your computer would recognize only your voice. And let you in. See **speech synthesis**.


video disk

viewdata

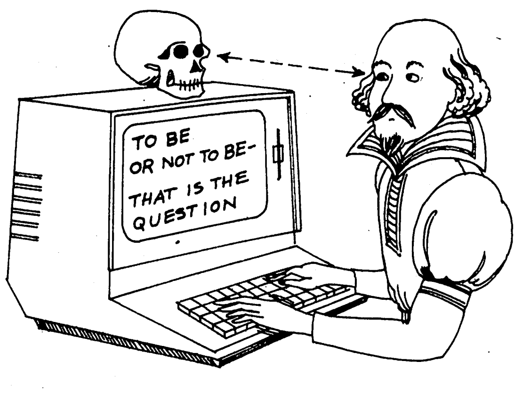
## W Z

wand

**wand** (wond): an object like a thick crayon, used for reading printed "bar codes" (stripes). The wand is connected to a computer. As you pass the wand over a bar code, it "sees" the different widths of the stripes. The computer then checks its memory to find out what each stripe stands for. See **bar code, optical character recognition, magnetic ink character recognition.**

**word** (wûrd): **1.** the number of electric pulses ("bits") that a computer treats as one item. In this sense, a computer word is like a word in human language. When your eye looks at this sentence, you read "words," not "w," plus "o," plus "r," "d," "s." Computers don't stop over a single electric pulse, either. Instead, they "read" each set of pulses (on, on, on, on, on, on, off, on) as a unit. All the words in one computer are the same length. In an 8-bit computer, all words are 8 bits long. There are also 16-bit computers, and computers that handle 32-bit words. See **bit, byte, nibble.** **2.** in LOGO, any string of letters or numbers that you type without a space between them. A LOGO word can even be one letter. See **string** (definition 2).

**word processing** (wûrd' pros' əs' ing): the use of a computer to store poems, plays, letters, and any other "text" that you write. A word processing program usually includes "editing" steps. These allow you to change, insert, delete, and move words around. If you want a final copy of what you write, you need a printer attachment for your computer. See **edit** (definition 2), **printer**.



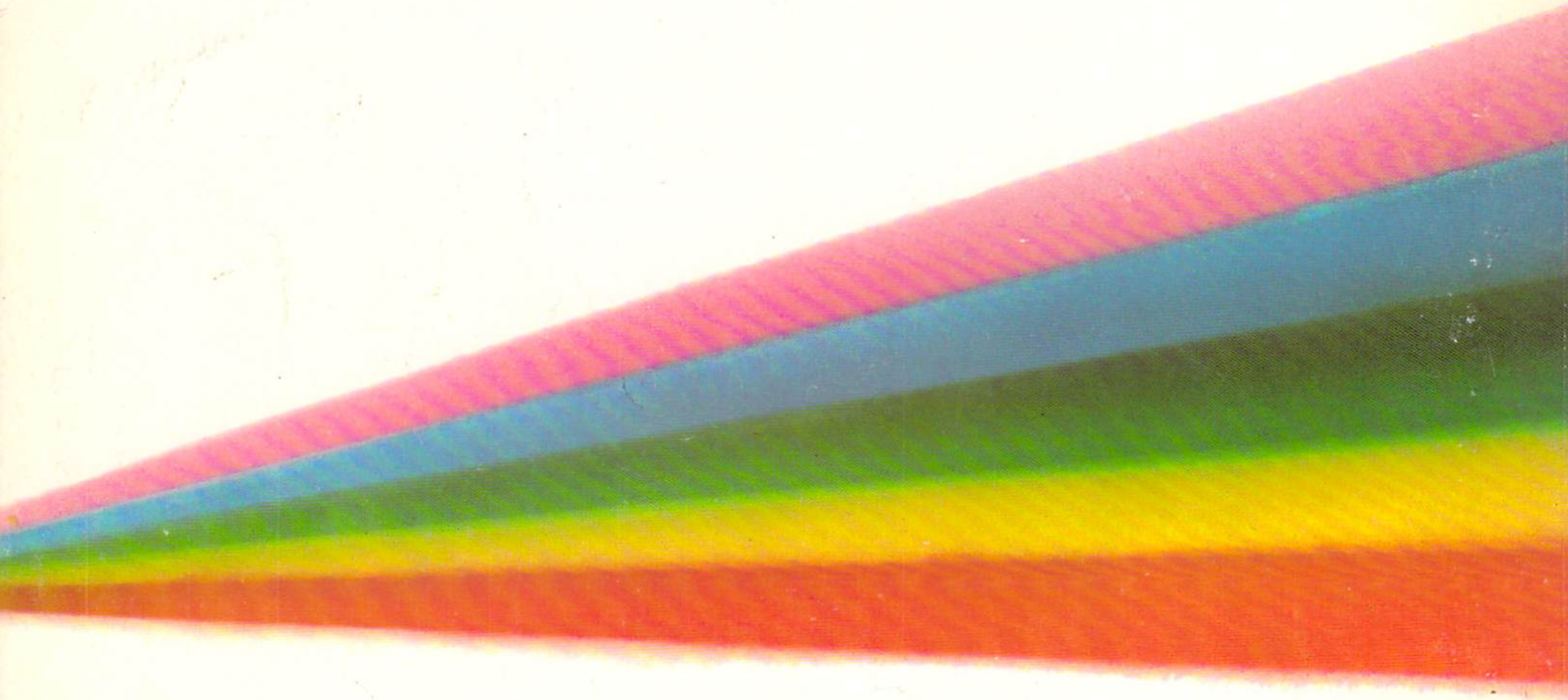
word processing

**workspace** (wûrk' spās): in LOGO, your computer's memory of what you type or "draw" while you're at the keyboard. If you turn off the computer, or type GOODBYE, that memory ("workspace") disappears. But there are ways to save workspace information. See **SAVE** and **SAVEPICT**. And **memory,** too.

**write** (rīt): **1.** for a computer, to put data into its storage (for example, onto a tape). See **read.** **2.** for a human, to invent a set of instructions (a program) for a computer.

**zero-bit** (zir' ō bit'), **0-bit**: a place-holder; a tiny pause between electric pulses that a computer "reads." A zero-bit is a partner to a "one-bit" (1-bit). Together, these two bits form the basis of "machine language." That's the code inside a computer. See **bit, one-bit, byte, binary system, language.**

0-590-94456-8



SCHOLASTIC INC.